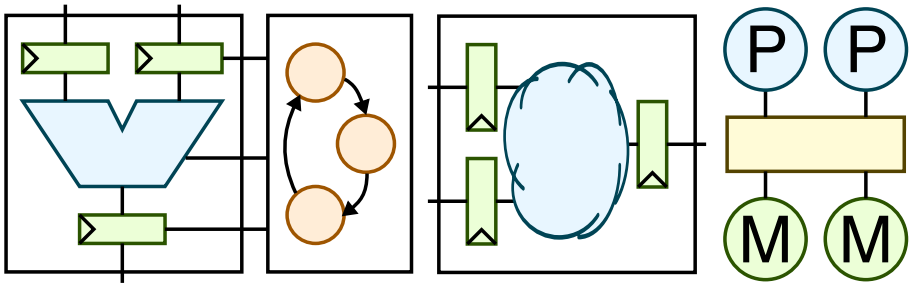
ECE 5745 Complex Digital ASIC Design
Topic 5: Automated Design Methodologies

Christopher Batten

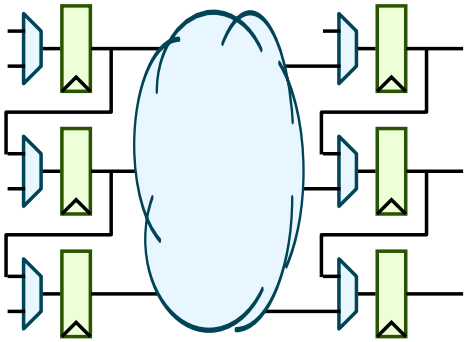
School of Electrical and Computer Engineering
Cornell University

http://www.csl.cornell.edu/courses/ece5745

Part 1: ASIC Design Overview



Topic 1
Hardware Description Languages

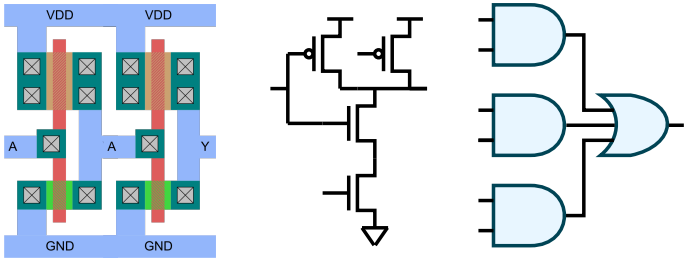


Topic 8
Testing and Verification

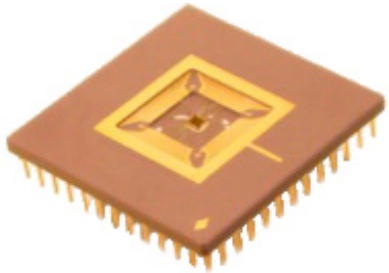
Topic 4
Full-Custom Design Methodology

Topic 6
Closing the Gap

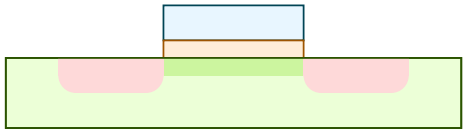
Topic 5
Automated Design Methodologies



Topic 3
CMOS Circuits

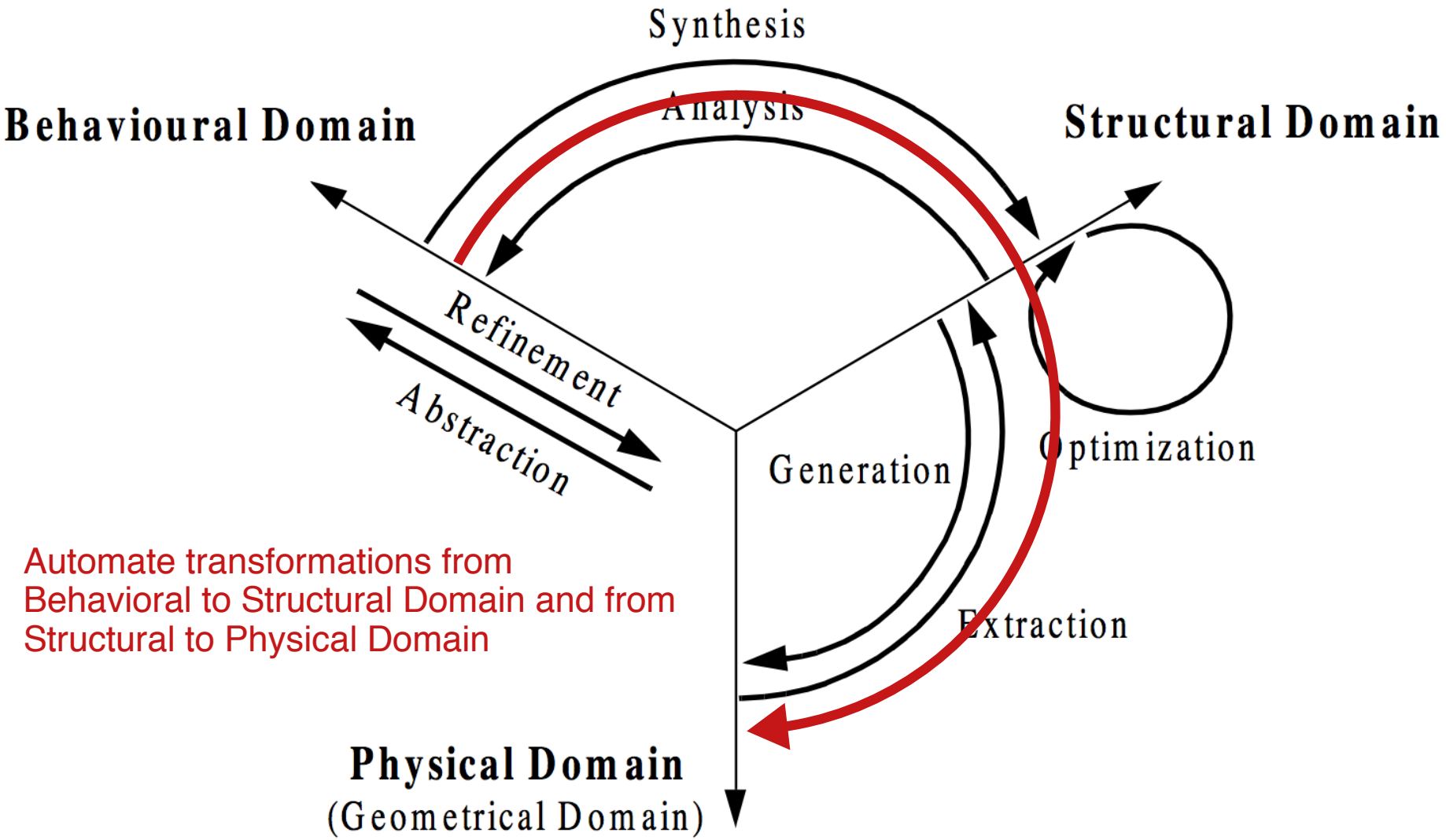


Topic 7
Clocking, Power Distribution, Packaging, and I/O



Topic 2
CMOS Devices

Automated Design Methodologies



Automate transformations from Behavioral to Structural Domain and from Structural to Physical Domain

Adapted from [Ellervee'04]

Design Principles in Automated Methodologies

- Modularity: Use modularity to enable mixing different custom and automated methodologies
- Hierarchy: Use hierarchy to more efficiently handle automatically transforming large designs
- Encapsulation: Automated methodologies have a significantly higher emphasis on encapsulation in all domains (behavoral, structural, physical) at all levels of abstraction: architecture-, register-transfer-, and gate-level
- Regularity: Use regularity to create automated tools to generate structures like datapaths and memories
- Extensibility: Automated methodologies enable more highly parameterized and flexible implementations improving extensibility

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

Gate-Array-Based Design

Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

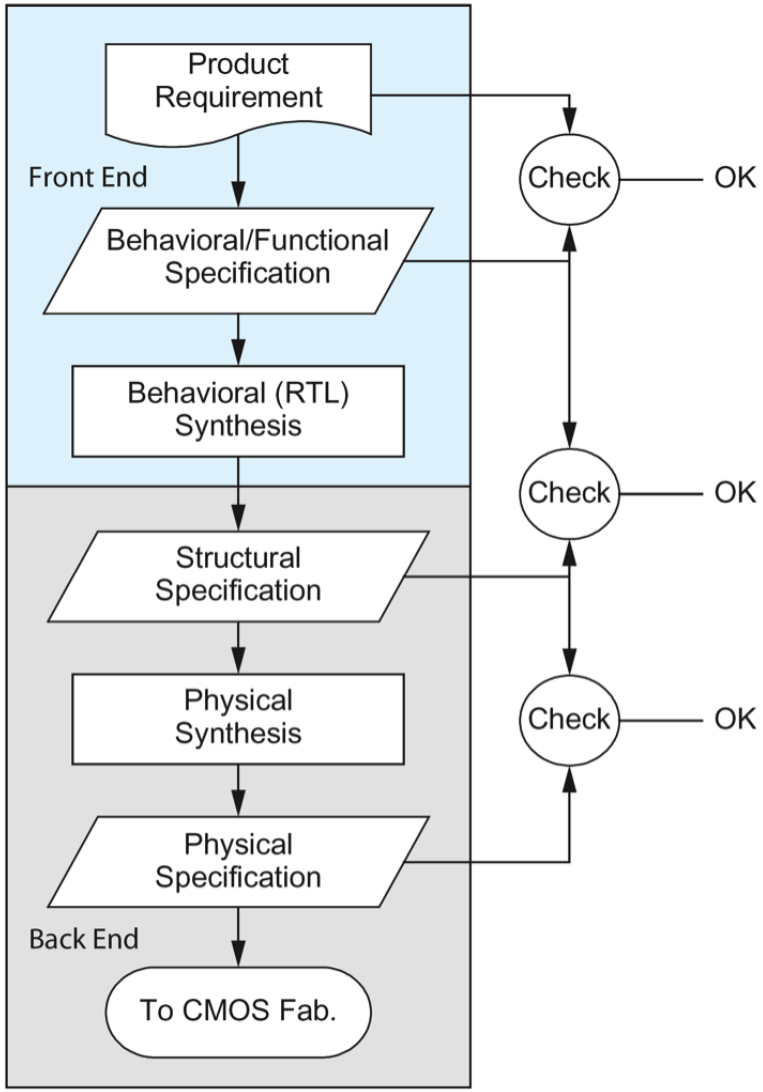
Gate-Array-Based Design

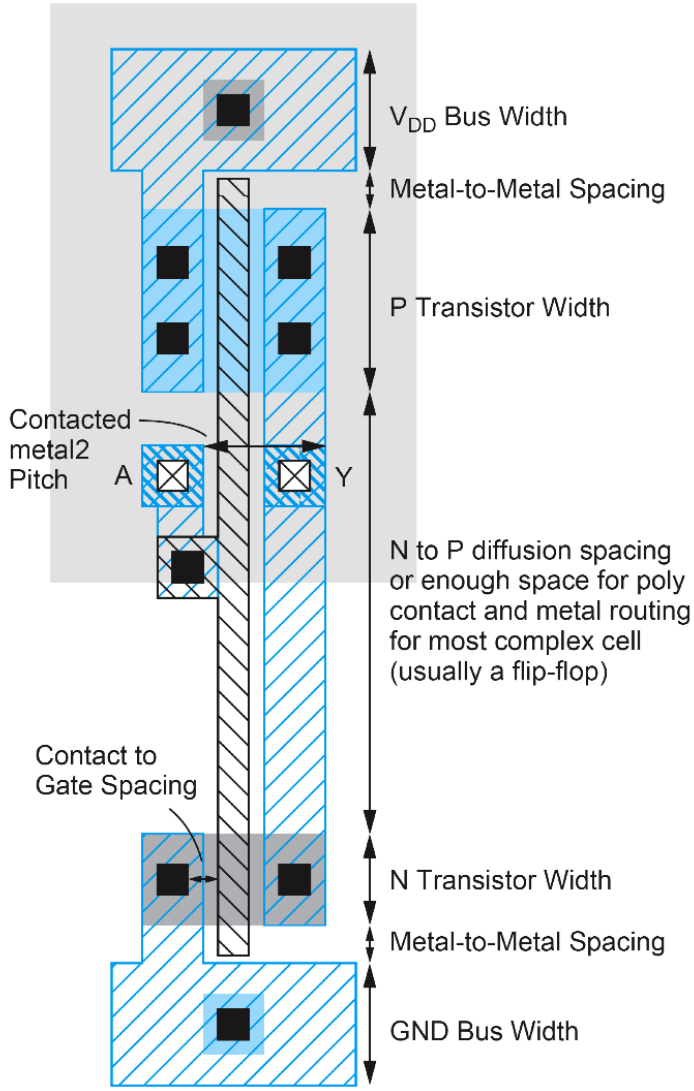
Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

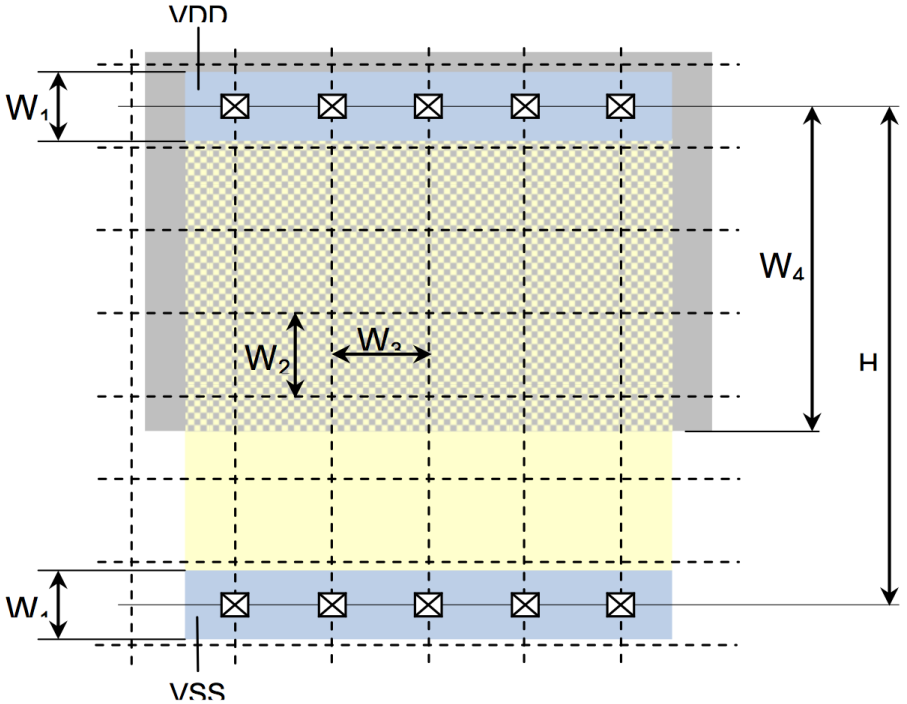
Standard-Cell-Based Design

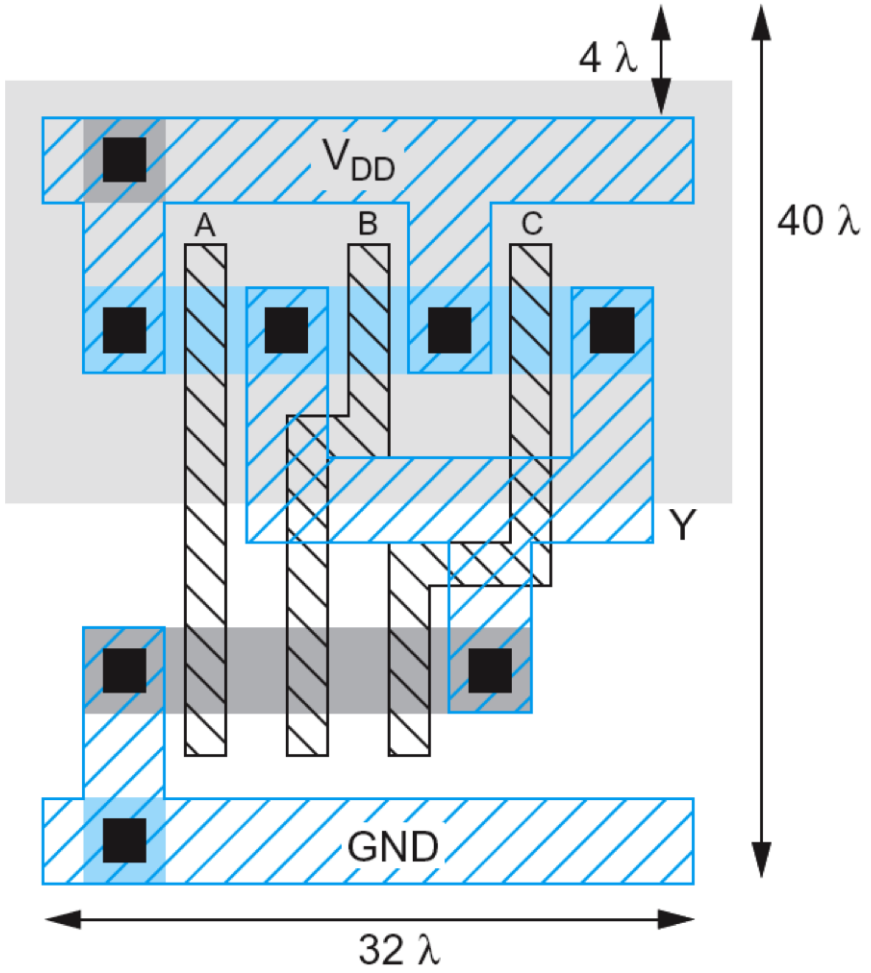




Adapted from [Weste'11]

Example Standard Cell

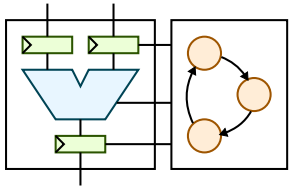




Adapted from [SAED'11,Weste'11]

Standard-Cell-Based Flow CAD Algorithms

Topic 12
Synthesis Algorithms



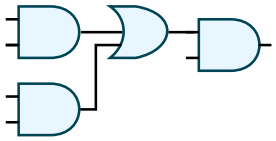
RTL to Logic Synthesis

x = a'bc + a'bc'
y = b'c' + ab' + ac

Technology Independent Synthesis

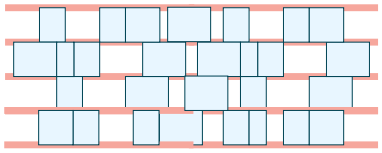
x = a'b
y = b'c' + ac

Technology Dependent Synthesis

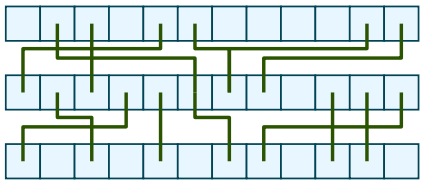


Topic 13
Physical Design Automation

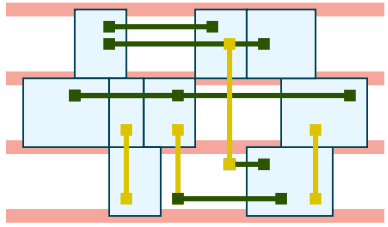
Placement



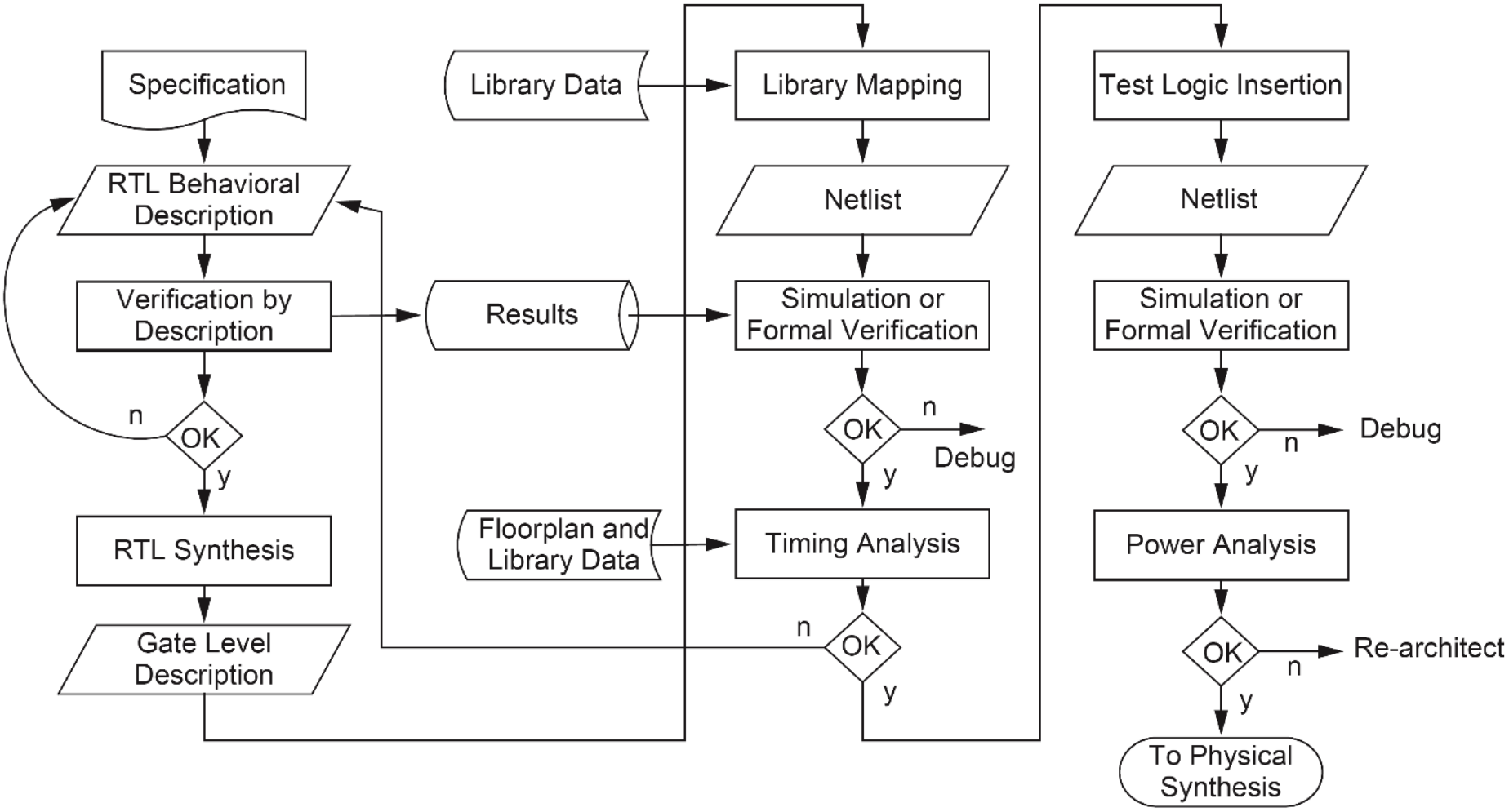
Global Routing



Detailed Routing

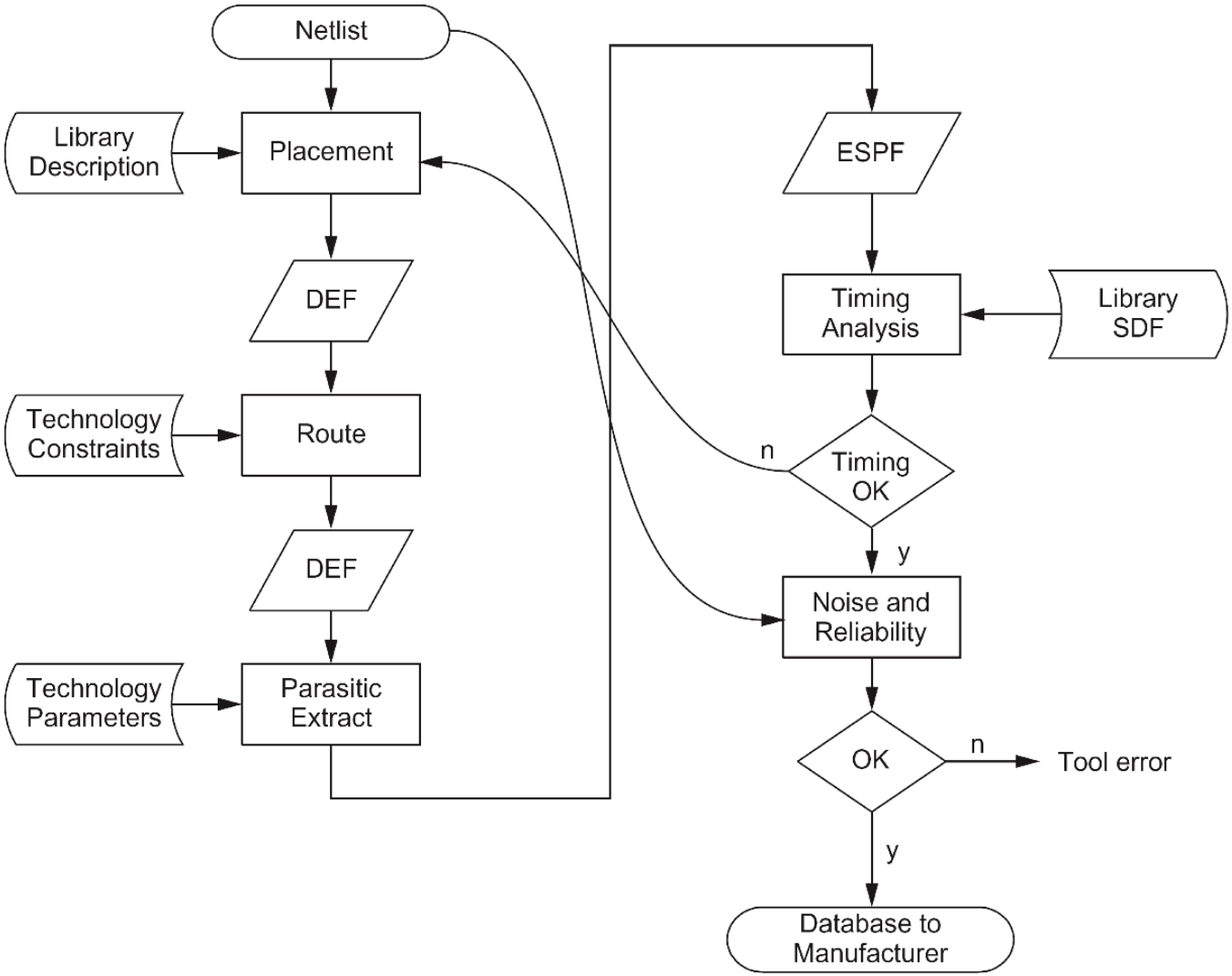


Standard-Cell-Based Front-End Flow



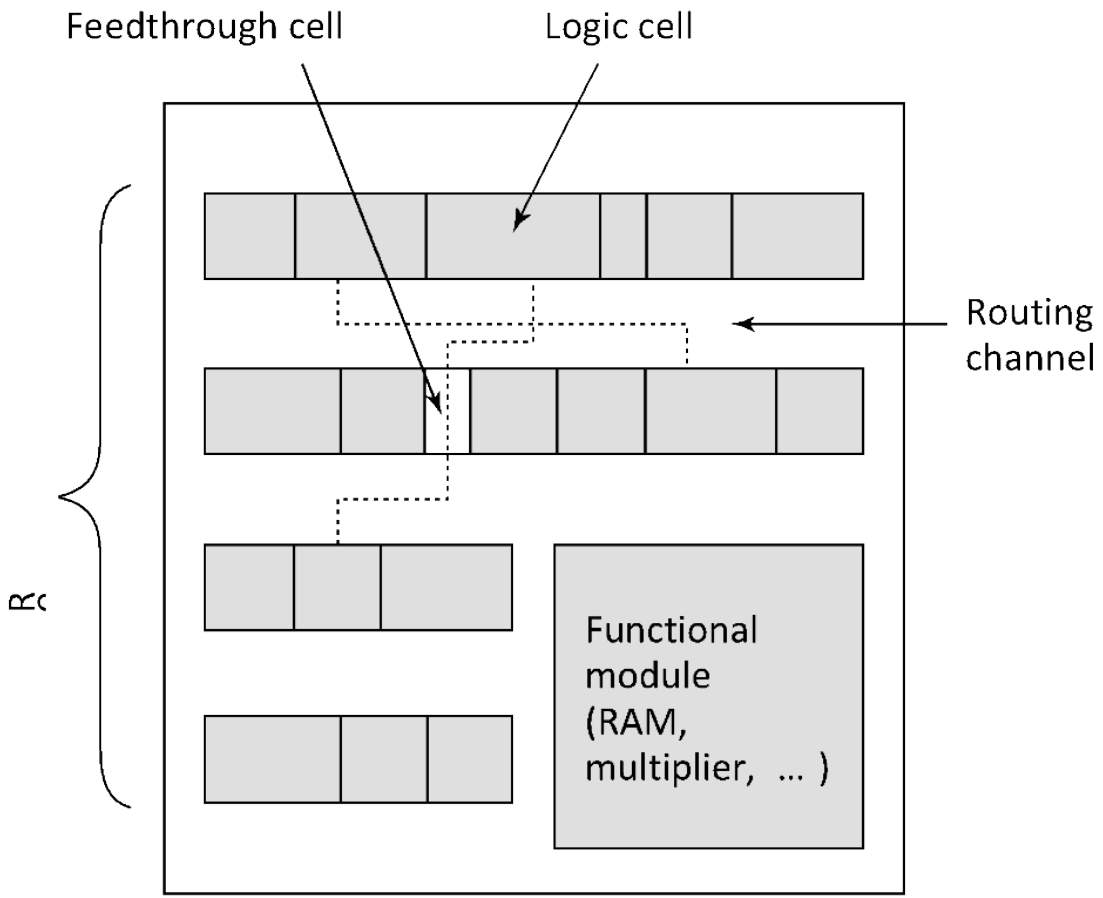
Adapted from [Weste'11]

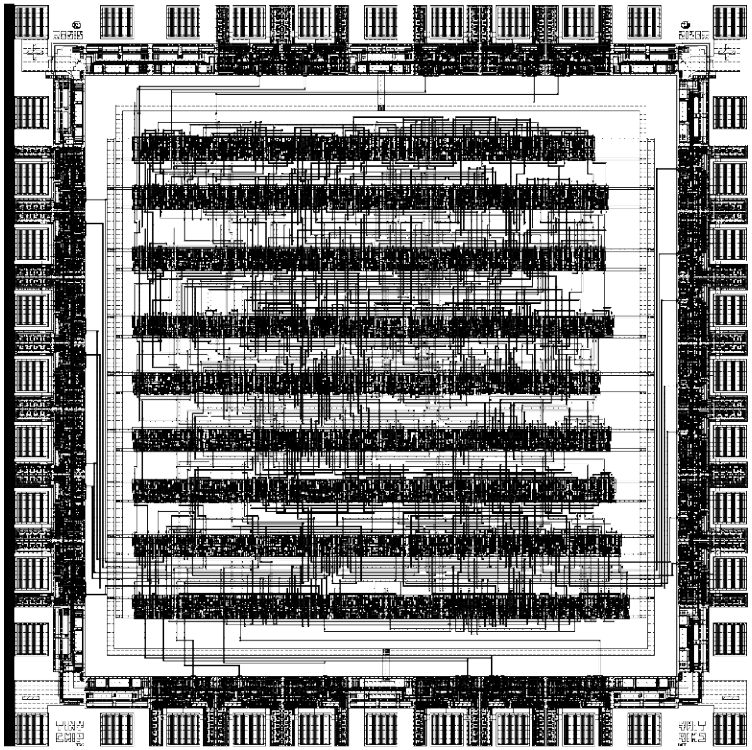
Standard-Cell-Based Back-End Flow



Adapted from [Weste'11]

Older Standard-Cell ASICS

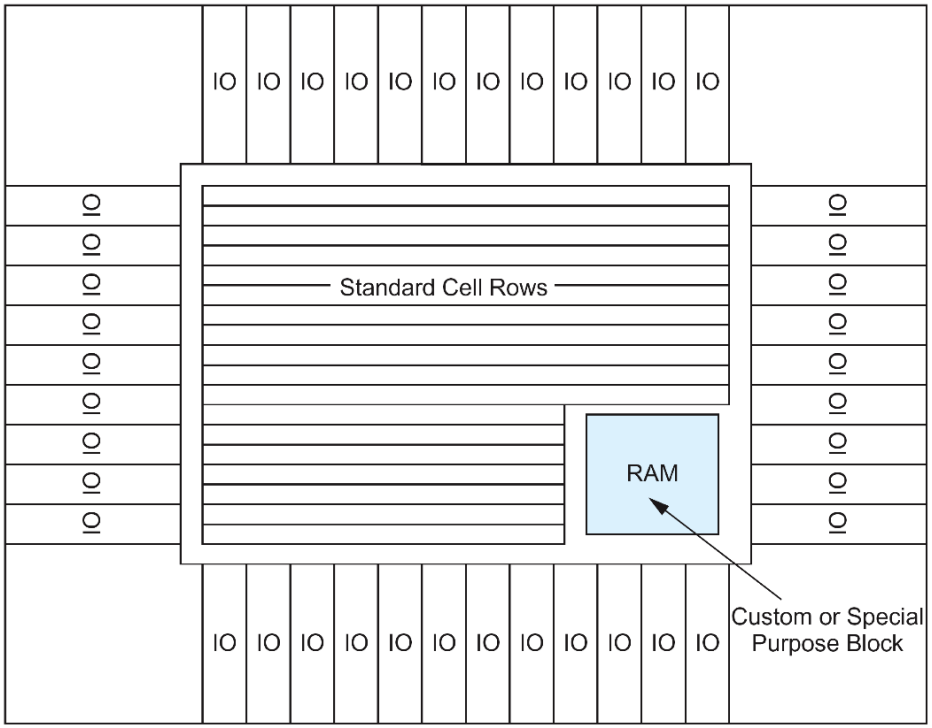


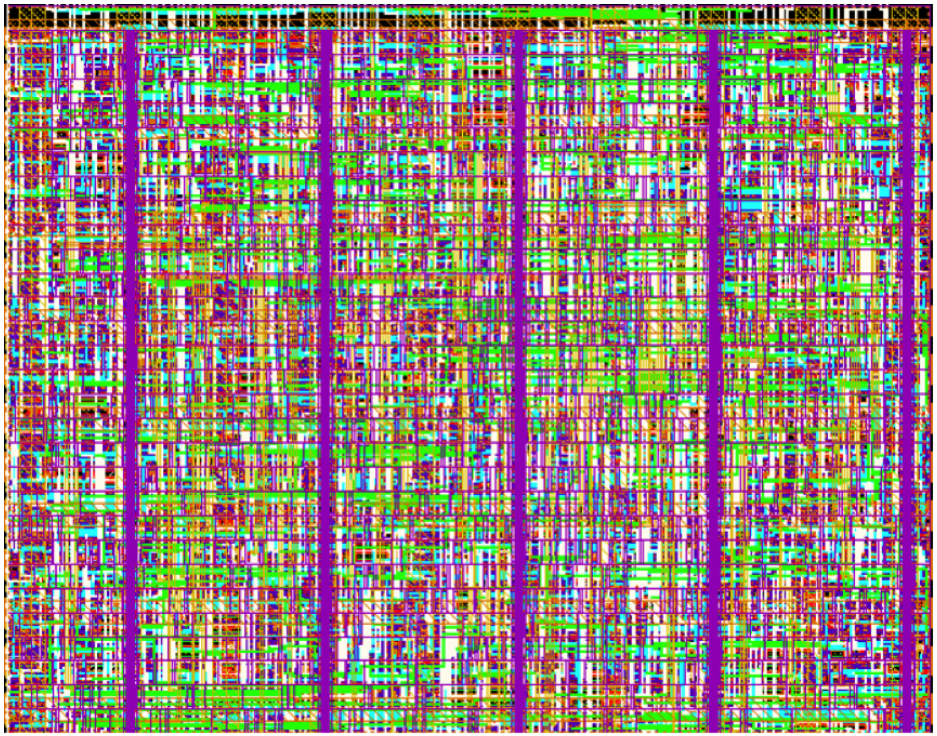


Limited metal layers require dedicated routing channels and feedthrough cells

Adapted from [Rabaey'02]

Modern Standard-Cell ASICS





Increasing number of metal layers allow cells to be hidden under interconnect

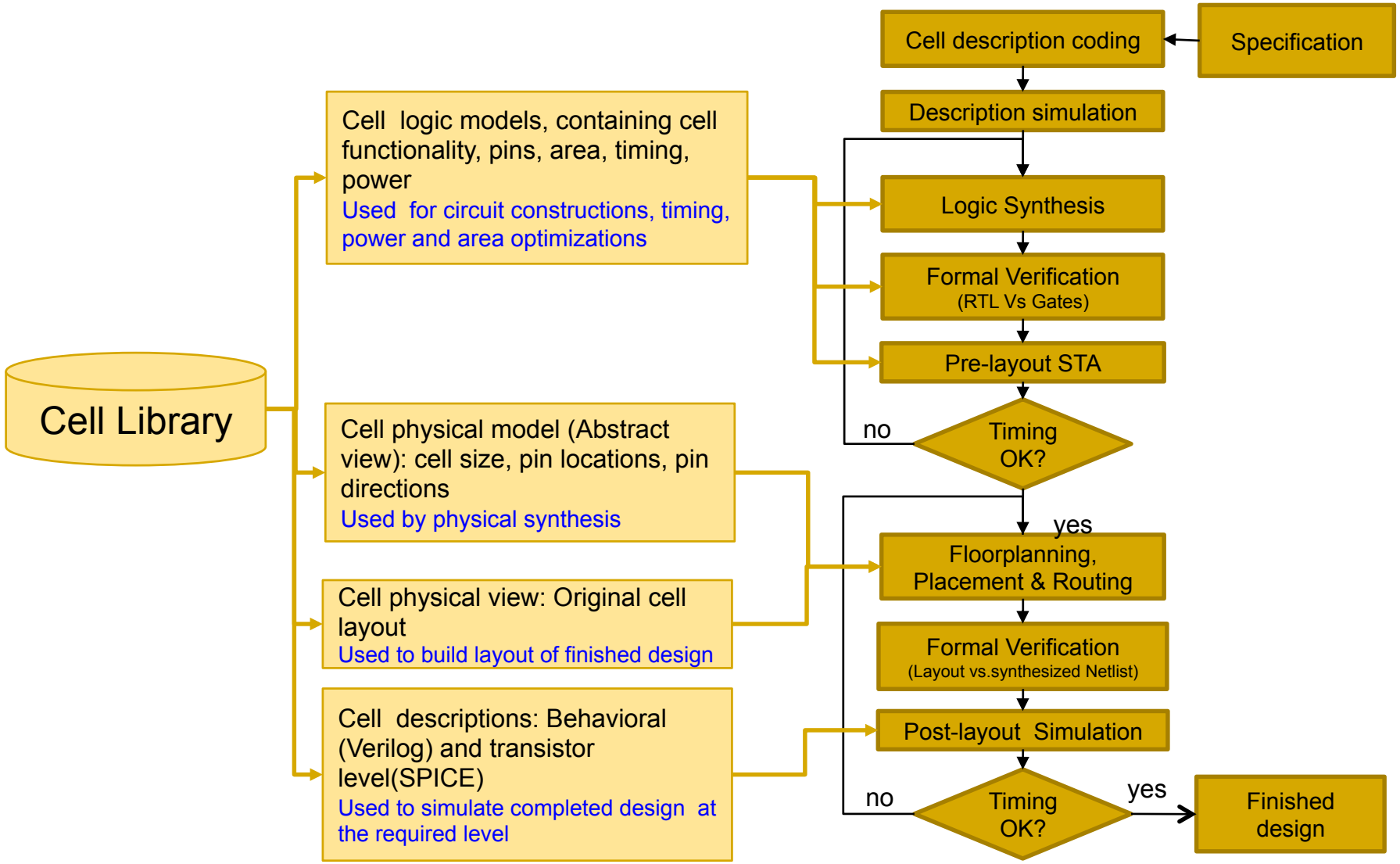
Adapted from [Weste'11,Rabaey'02]

Standard-Cell Libraries

Gate Type	Variations	Options
Inverter/Buffer/ Tristate Buffers		Wide range of power options, 1x, 2x, 4x, 8x, 16x, 32x, 64x minimum size inverter
NAND/AND	2–8 inputs	High, normal, low power
NOR/OR	2–8 inputs	High, normal, low power
XOR/XNOR		High, normal, low power
AOI/OAI	21, 22	High, normal, low power
Multiplexers	Inverting/noninverting	High, normal, low power
Adder/Half Adder		High, normal, low power
Latches		High, normal, low power
Flip-Flops	D, with and without synch/asynch set and reset, scan	High, normal, low power
I/O Pads	Input, output, tristate, bidirectional, boundary scan, slew rate limited, crystal oscillator	Various drive levels (1–16 mA) and logic levels

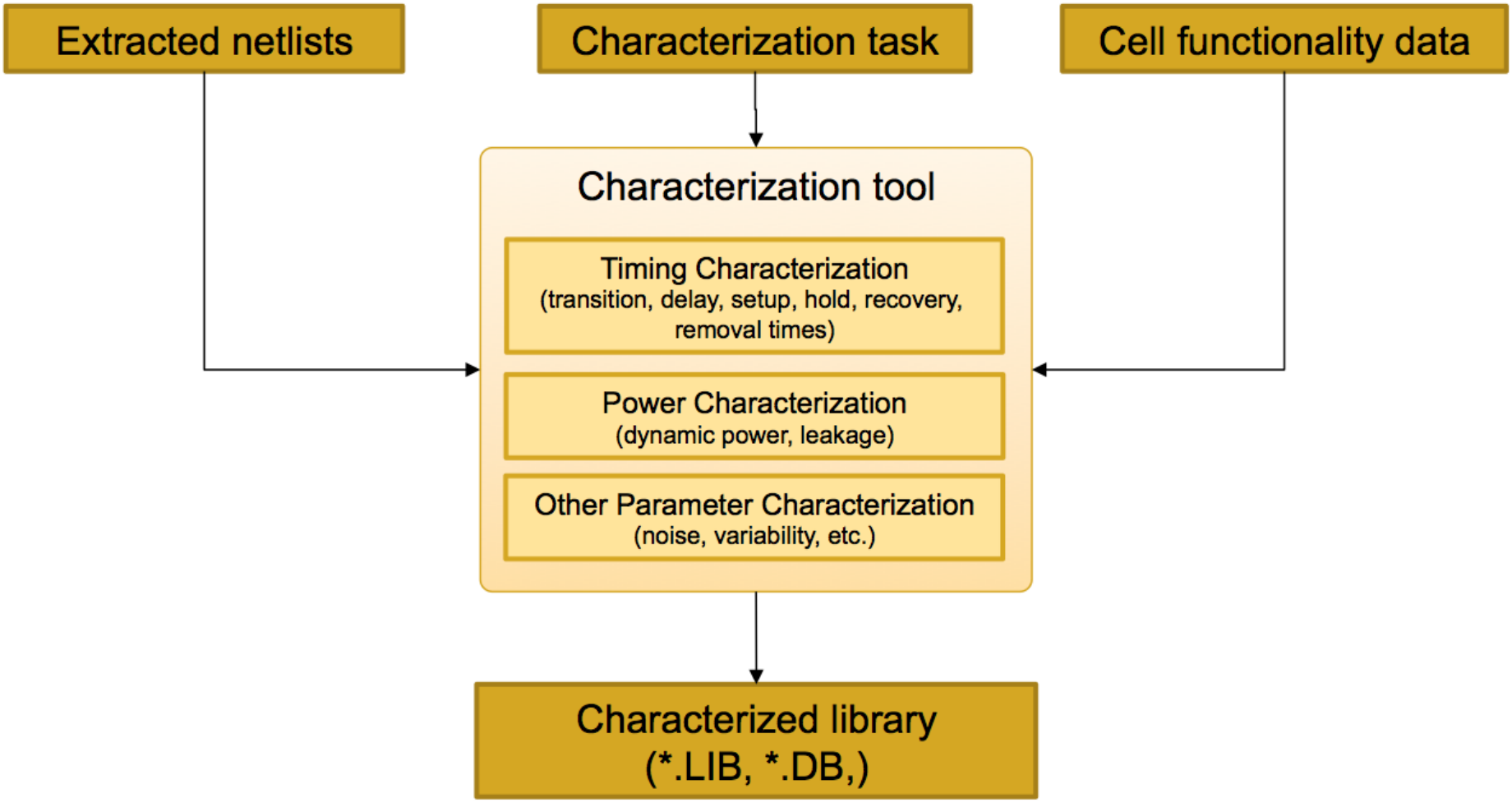
Adapted from [Weste'11]

Standard-Cell Libraries



Adapted from [Melikyan]

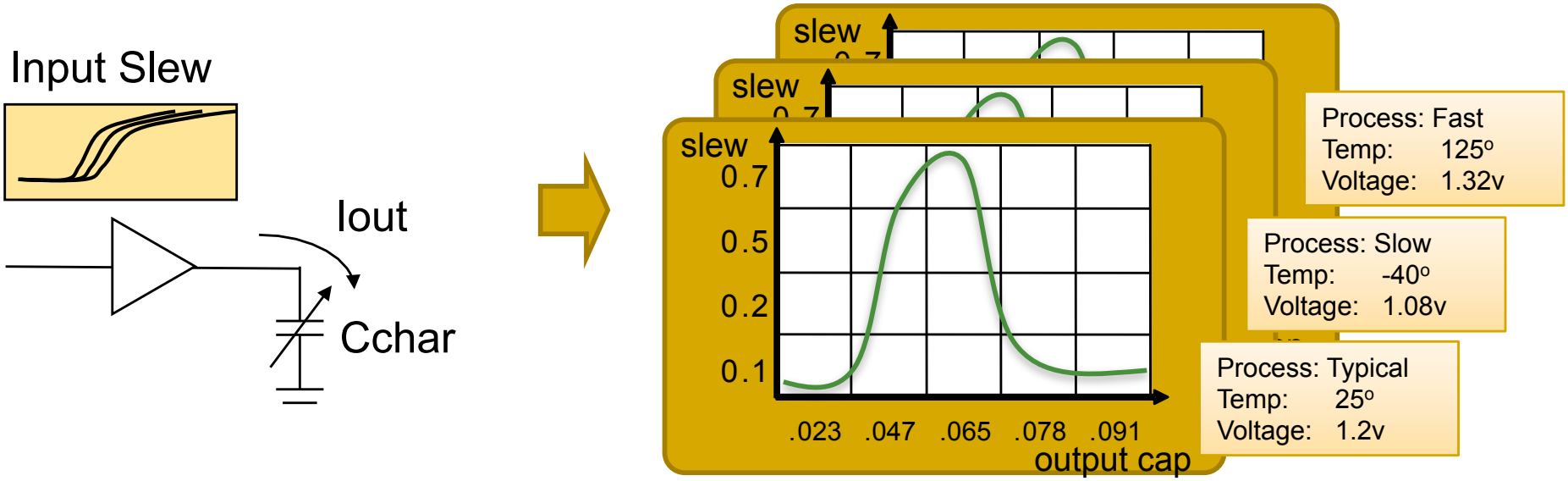
Standard-Cell Library Characterization



Adapted from [Melikyan]

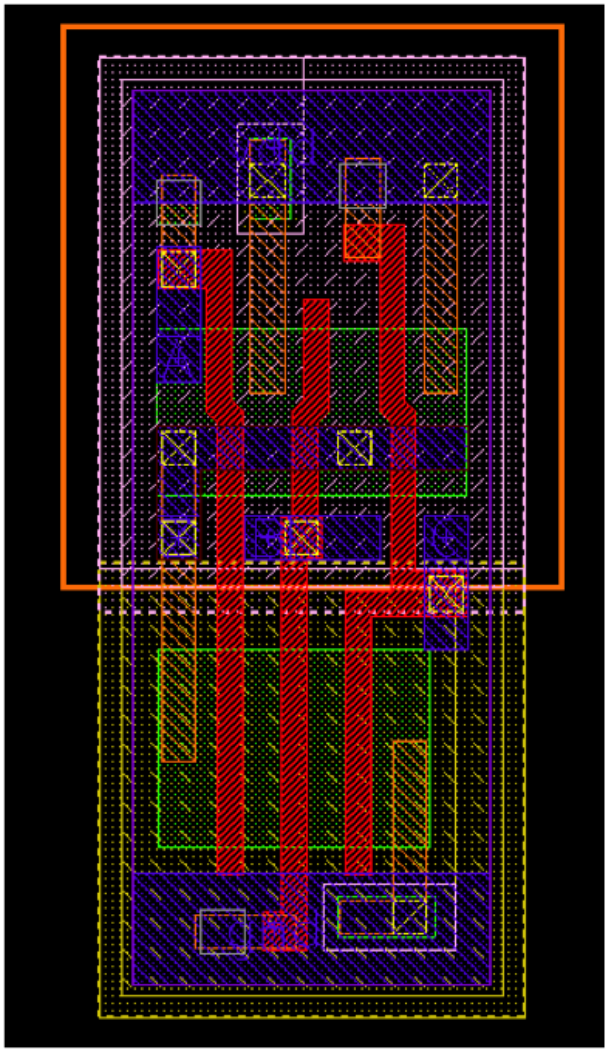
Standard-Cell Electrical Characterization

- Characterization computes cell parameter (e.g. delay, output current) depending on input variables: output load, input slew, etc.)
- Characterization is performed for various combinations of operating conditions: process, voltage, temperature (also called PVT corners)



Adapted from [Melikyan]

ST Microelectronics – 3-Input NAND Gate



Path	1.2V - 125°C	1.6V - 40°C
$In1—t_{pLH}$	$0.073+7.98C+0.317T$	$0.020+2.73C+0.253T$
$In1—t_{pHL}$	$0.069+8.43C+0.364T$	$0.018+2.14C+0.292T$
$In2—t_{pLH}$	$0.101+7.97C+0.318T$	$0.026+2.38C+0.255T$
$In2—t_{pHL}$	$0.097+8.42C+0.325T$	$0.023+2.14C+0.269T$
$In3—t_{pLH}$	$0.120+8.00C+0.318T$	$0.031+2.37C+0.258T$
$In3—t_{pHL}$	$0.110+8.41C+0.280T$	$0.027+2.15C+0.223T$

3-input NAND cell
(from ST Microelectronics):
C = Load capacitance
T = input rise/fall time

Adapted from [Rabaey'02]

Standard-Cell Electrical Characterization (.lib)

```
/* Characterization for a 3-input NAND gate */
cell ( NAND3X0 ) {

  /* Overall characterization */
  cell_footprint      : "nand3x0";
  area                : 7.3728;
  cell_leakage_power  : 9.151417e+04;

  /* Characterization for input pin IN1 */
  pin ( IN1 ) {
    direction : "input";

    /* Fixed input capacitance */
    capacitance : 2.190745;

    /* Transient capacitance values */
    fall_capacitance : 2.212771;
    rise_capacitance : 2.168719;
```

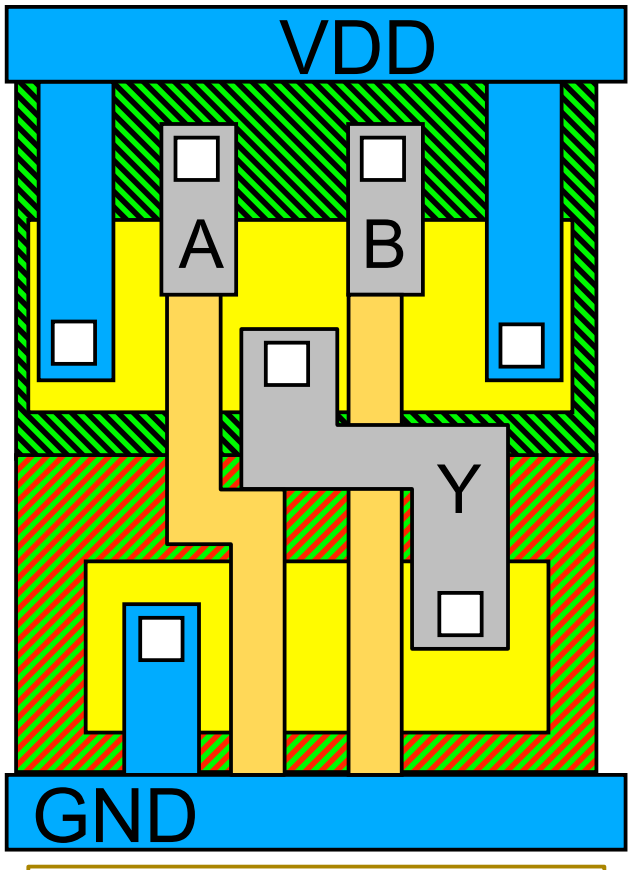
Standard-Cell Electrical Characterization (.lib)

```
cell ( NAND3X0 ) {
  pin ( IN1 ) {

    /* Short-circuit and internal switching
       power when IN2 and IN3 are zero */
    internal_power () {
      when : "!IN2&!IN3";

      /* 1D lookup tables to calculate power as
         function of input slew */
      rise_power ( "power_inputs_1" ) {
        index_1( " 0.0160000,  0.0320000,  0.0640000" );
        values(  "-1.2575404, -1.2594251, -1.2887053" );
      }
      fall_power ( "power_inputs_1" ) {
        index_1( " 0.0160000,  0.0320000,  0.0640000" );
        values(  " 1.9840914,  1.9791286,  2.0696119" );
      }
    }
```

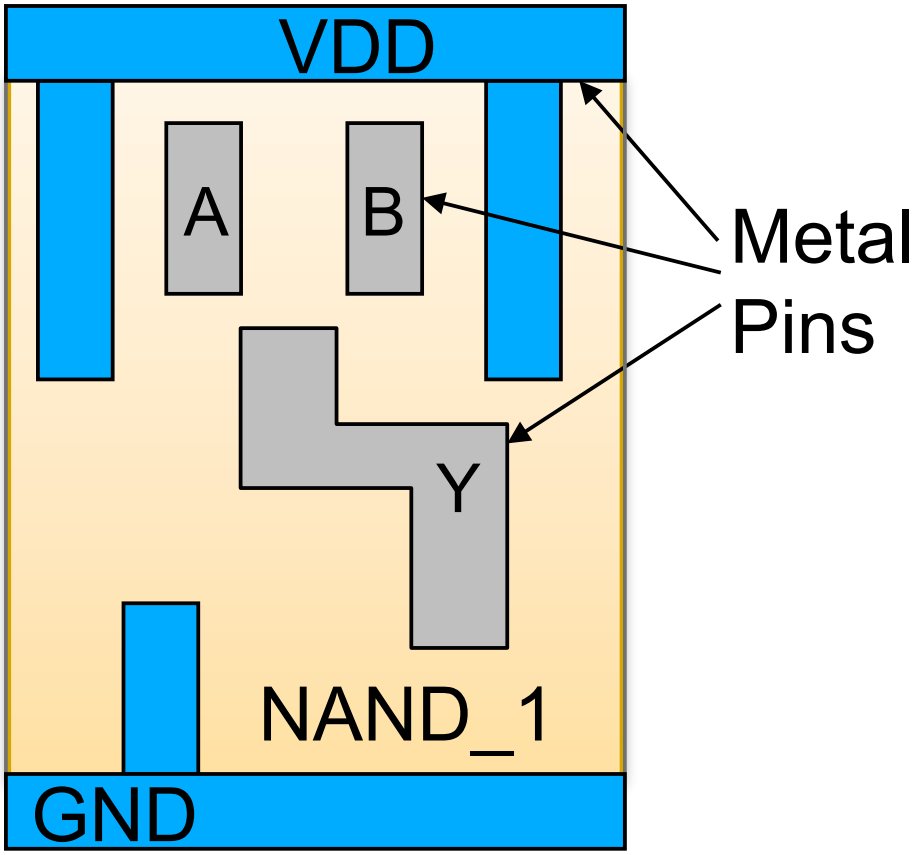
Standard-Cell Electrical Characterization (.lib)

```
cell ( NAND3X0 ) {
  pin ( QN ) {
    direction : "output";

    /* Boolean logic eq for QN as function of inputs */
    function : "(IN3*IN2*IN1)'";

    timing () {
      related_pin : "IN1";
      cell_rise ( "del_1_7_7" ) {

        index_1( "0.016, 0.032, 0.064" ); /* Input slew */
        index_2( "0.1,   3.75,  7.5"   ); /* Load cap */

        /* Lookup table to calculate delay as non-linear
           function of input signal slew and load cap */
        values( "0.0178632, 0.0275957, 0.0374970", \
                "0.0215562, 0.0316225, 0.0414275", \
                "0.0261721, 0.0387623, 0.0496870" )
      }
```

Standard-Cell Physical Characterization



Layout View



Abstract Physical View

Adapted from [Melikyan]

SAED 90 nm Library – NAND Gate Data Sheet

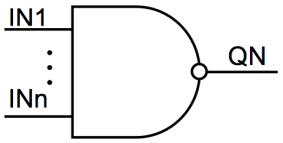


Figure 10.6. Logic Symbol of NAND

Table 10.11. NAND Truth Table (n=2,3,4)

IN1	IN2	. . .	INn	QN
0	X	. . .	X	1
X	0	. . .	X	1
. . .	. . .	. . .	. . .	1
X	X	. . .	0	1
1	1	1	1	0

Cell Name	Operating Conditions: VDD=1.2 V DC, Temp=25 Deg.C, Operating Frequency: Freq=300 MHz, Capacitive Standard Load: Csl=13 fF				Area
	Cload	Prop Delay (Avg)	Power		
			Leakage (VDD=1.2 V DC, Temp=25 Dec.C)	Dynamic	
		ps	nW	nW/MHz	(um^2)
NAND2X0	0.5 x Csl	140	38	3583	5.5296
NAND2X1	1 x Csl	132	78	5208	5.5296
NAND2X2	2 x Csl	126	157	9191	9.2160
NAND2X4	4 x Csl	125	314	17902	14.7456
NAND3X0	0.5 x Csl	128	91	5331	7.3728
NAND3X1	1 x Csl	192	102	12200	11.9808
NAND3X2	2 x Csl	212	155	19526	12.9024
NAND3X4	4 x Csl	241	260	44937	15.6672
NAND4X0	0.5 x Csl	147	106	5357	8.2944
NAND4X1	1 x Csl	178	161	15214	12.9024

Adapted from [SAED'11]

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

Gate-Array-Based Design

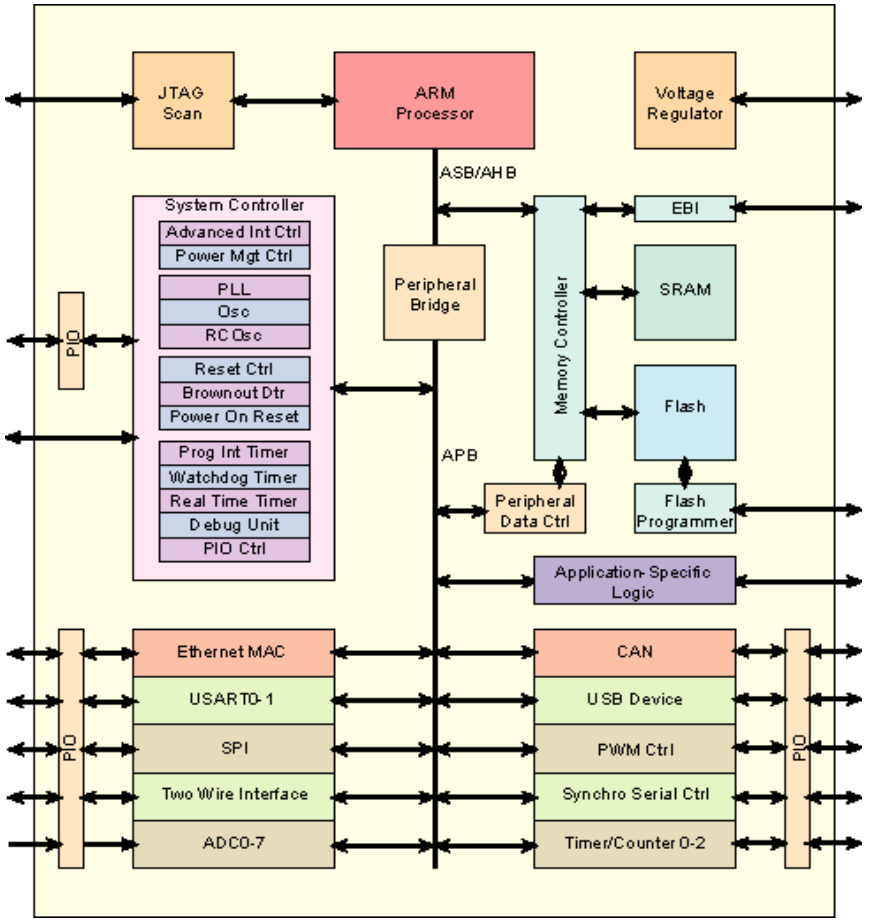
Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

System-on-Chip (SoC)

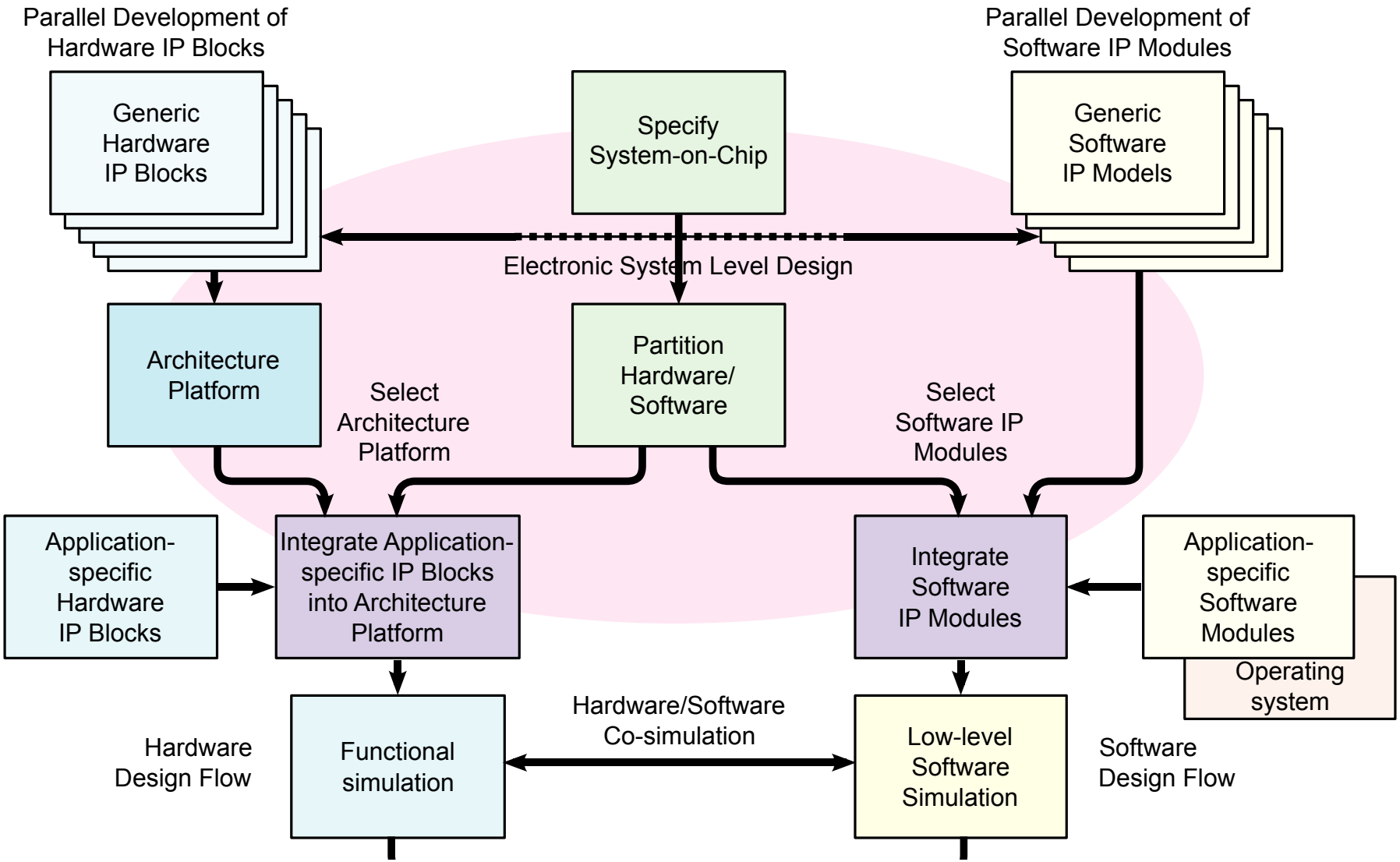
- Brings together: standard cell blocks, custom analog blocks, processor cores, memory blocks
- Standardized on-chip buses (or hierarchical interconnect) permit "easy" integration of many blocks.
  - Ex: AMBA, Sonics, …
- "IP Block" business model: Hard- or soft-cores available from third party designers.
- ARM, inc. is the shining example. Hard- and "synthesizable" RISC processors.
- ARM and other companies provide, Ethernet, USB controllers, analog functions, memory blocks, …



- Pre-verified block designs, standard bus interfaces (or adapters) ease integration - lower NREs, shorten TTM.

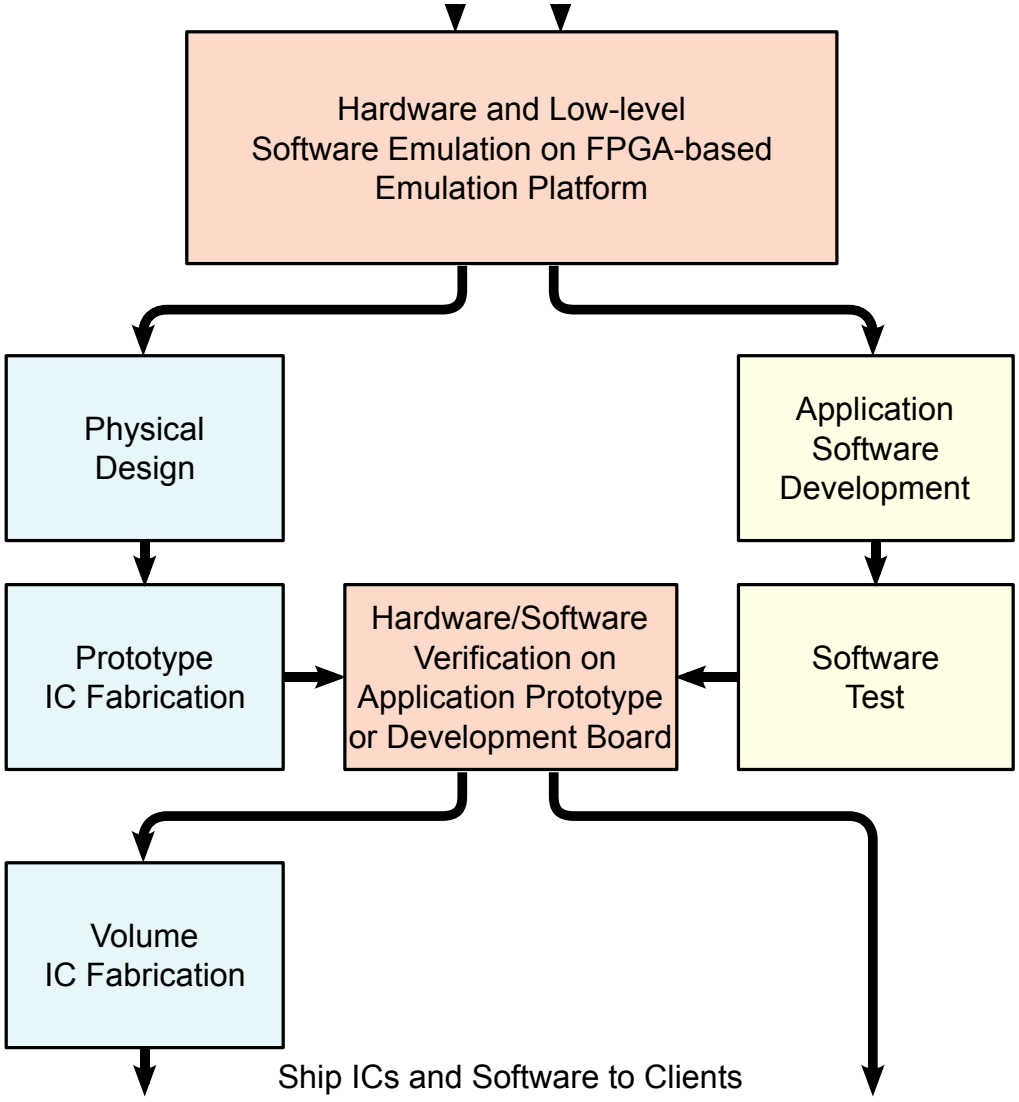
Adapted from [Asanovic'11]

SoC Hardware/Software Co-Design



Adapted from [Wikipedia]

SoC Hardware/Software Co-Design

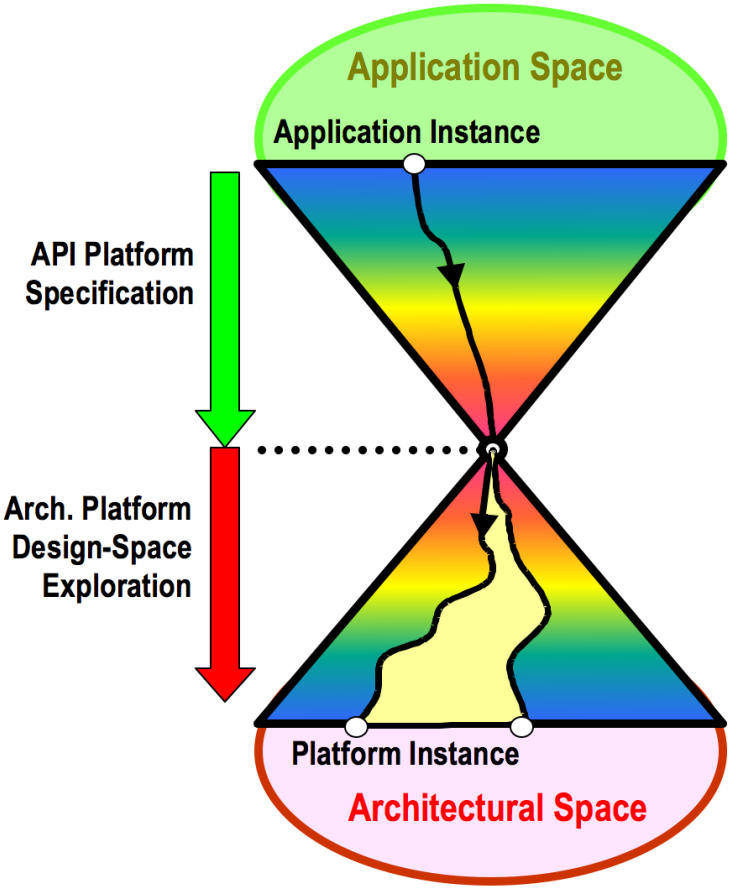


Adapted from [Wikipedia]

SoC Platform-Based Design

"Only the consumer gets freedom of choice;
designers need freedom *from* choice." – Orfali

- A platform is a restriction on the space of possible implementation choices, providing well-defined abstraction of the underlying technology for the app developer
- New platforms defined at architecture/ microarchitecture boundary
- Key to such approaches is the representation of communication in the platform model



Adapted from [ASV'01,Rabaey'02]

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

Gate-Array-Based Design

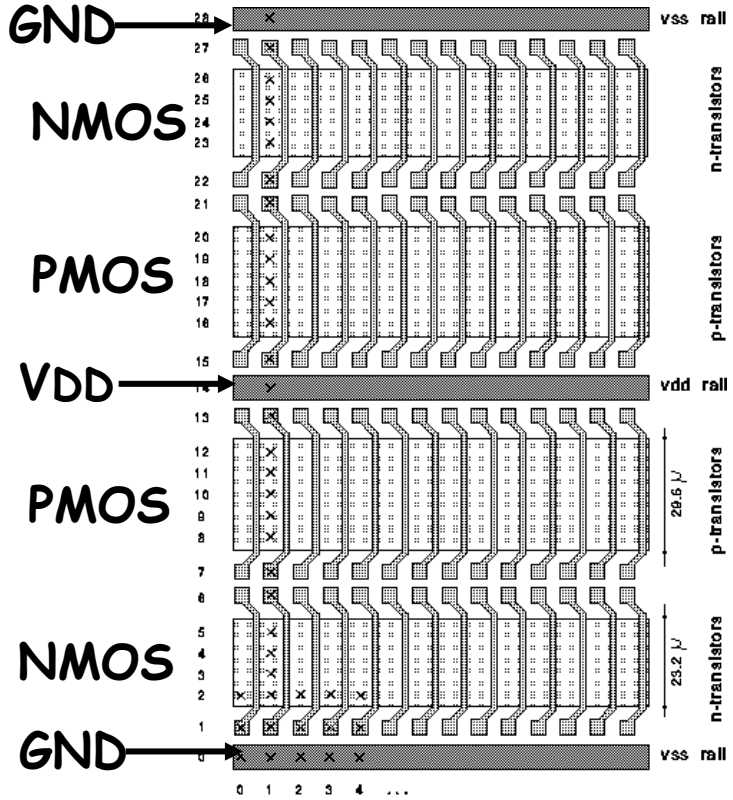
Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

Gate-Array-Based Design

- Can cut mask costs by prefabricating arrays of transistors on wafers
- Only customize metal layer for each design



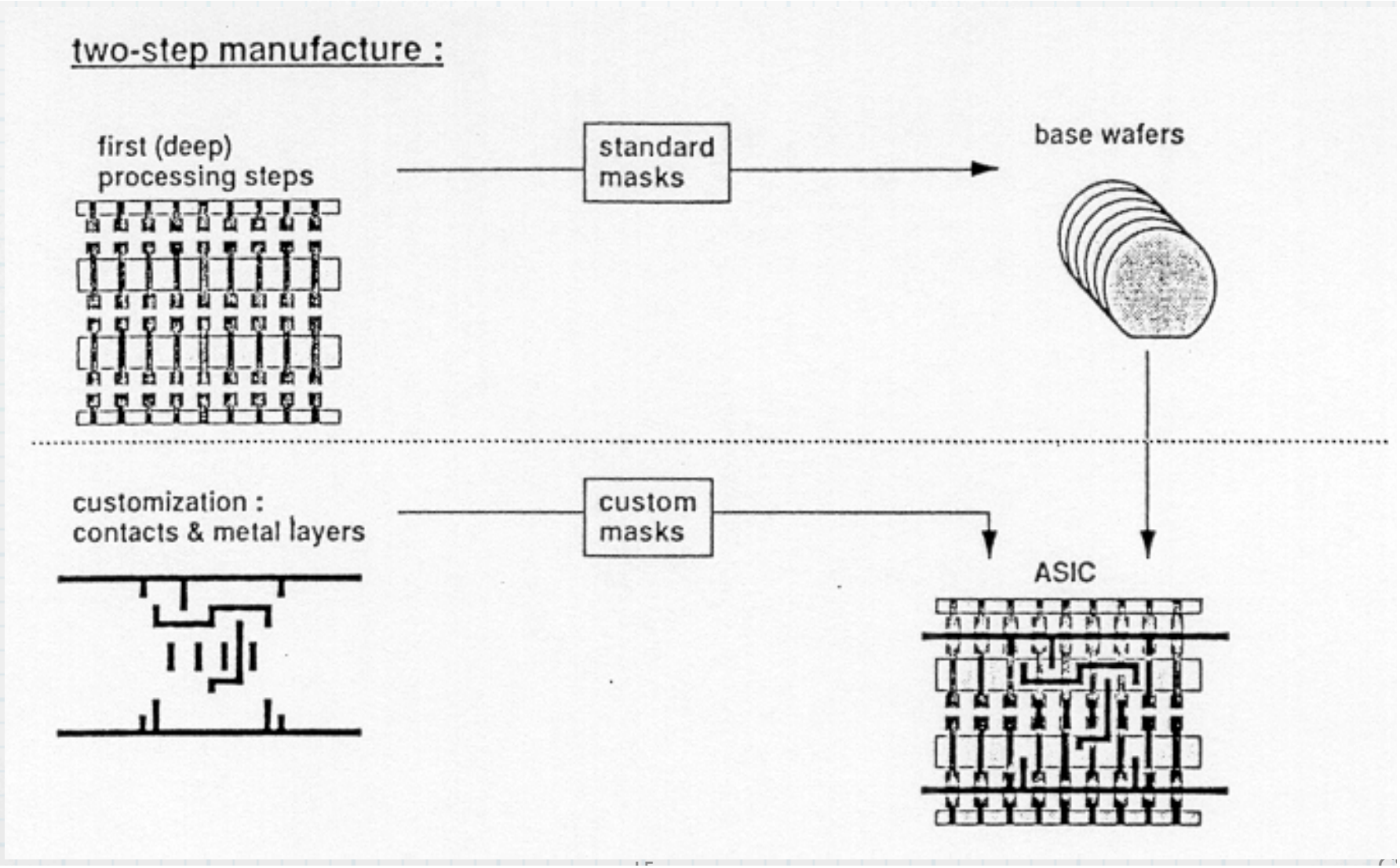
- Fixed-size unit transistors
- Metal connections personalize design

Two kinds:

- Channeled Gate Arrays
  - Leave space between rows of transistors for routing
- Sea-of-Gates
  - Route over the top of unused transistors

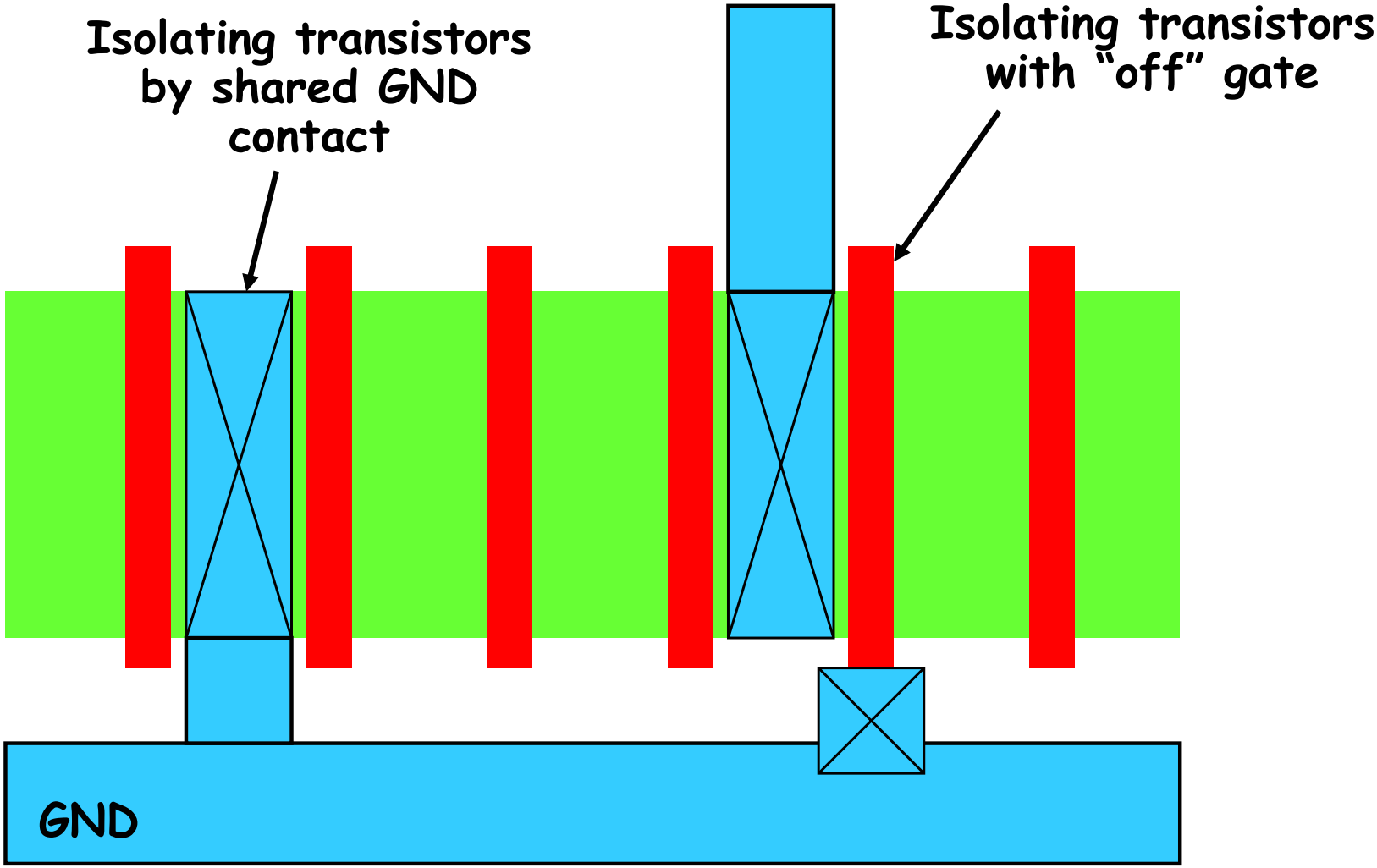
Adapted from [Terman'02]

Gate-Array Fabrication Process



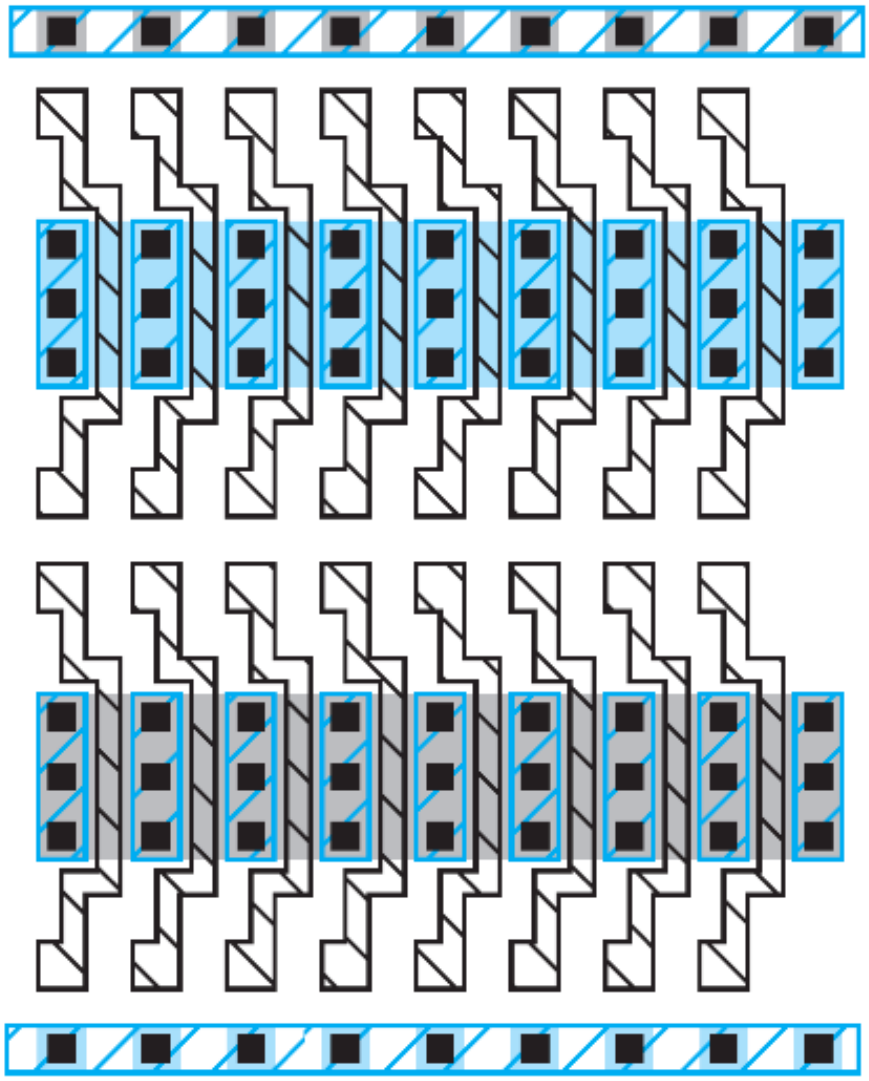
Adapted from [Asanovic'11]

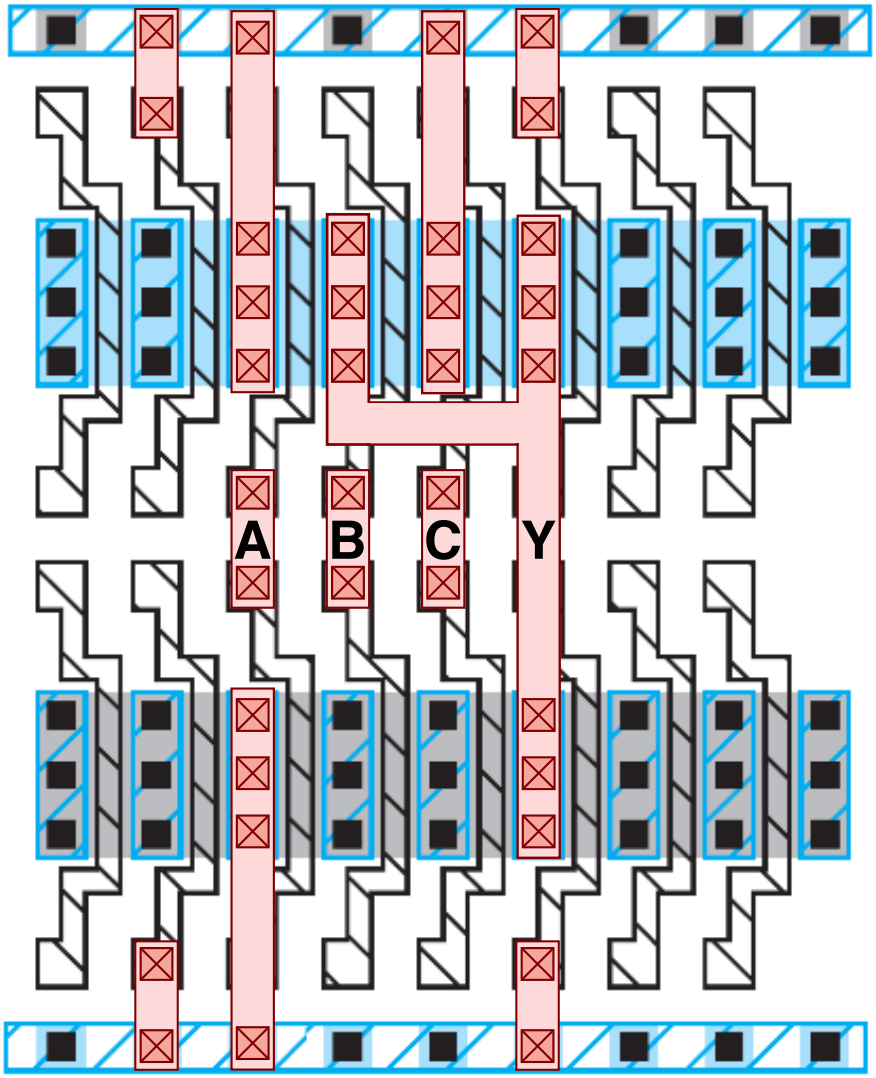
Gate-Array Personalization



Adapted from [Terman'02]

Gate-Array Example – 3-input NAND Gate





Adapted from [Weste'11]

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

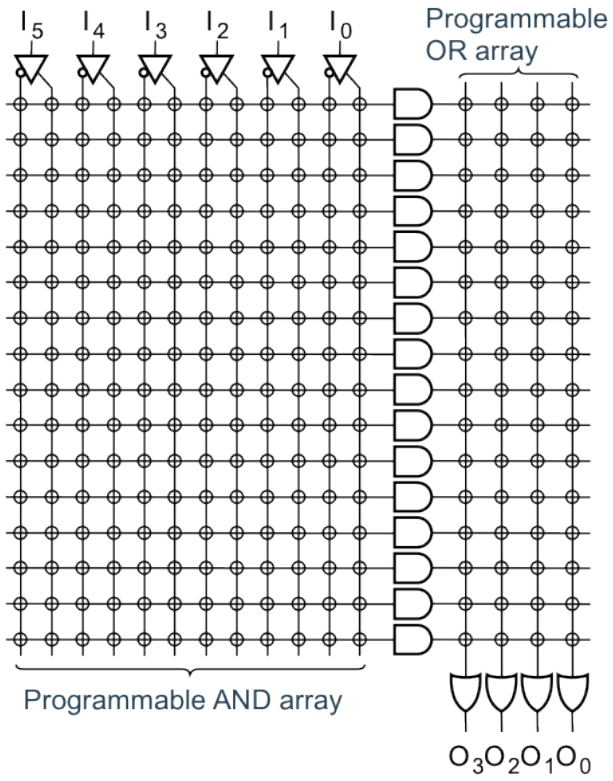
Gate-Array-Based Design

Programmable Logic-Based Design

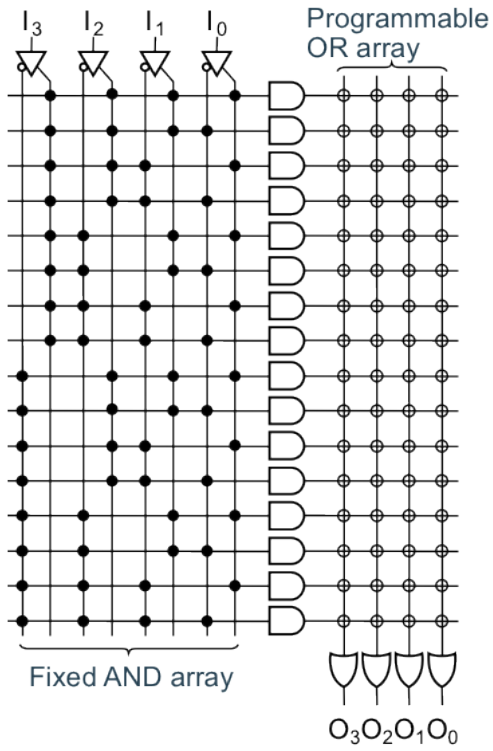
Reprogrammable Logic-Based Design

Comparison and Hybrids

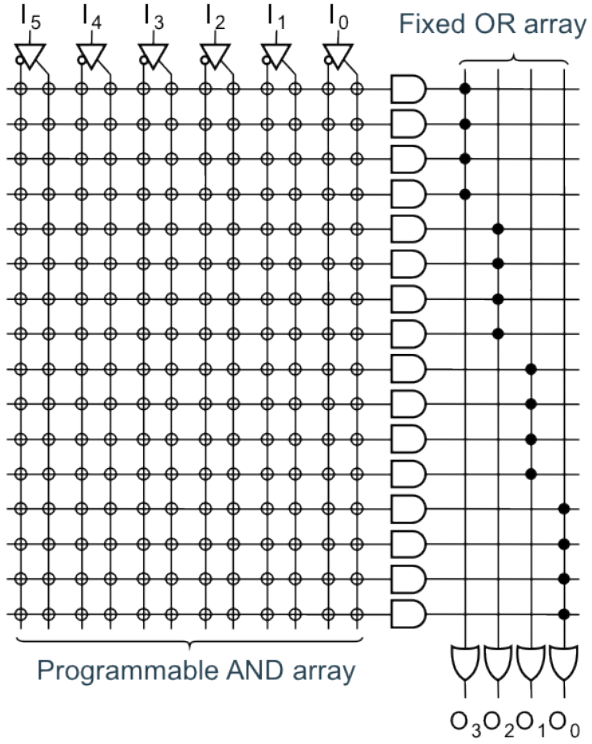
Array-Based Programmable Logic



PLA



PROM



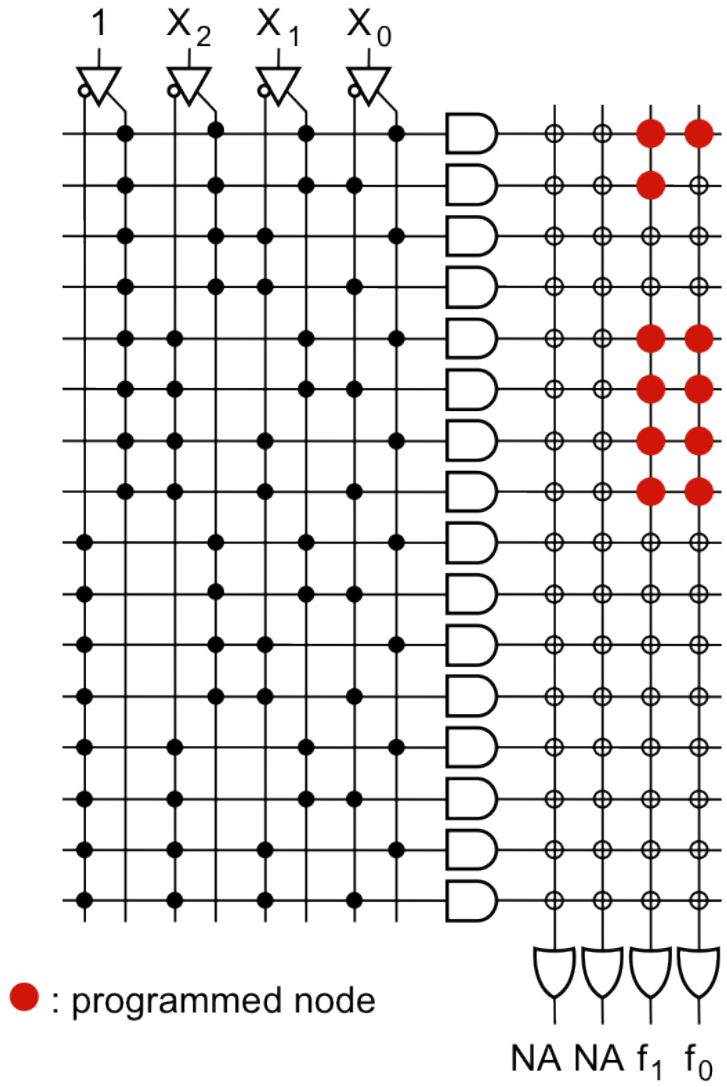
PAL

Indicates programmable connection

Indicates fixed connection

Adapted from [Rabaey'02]

Programming a PROM

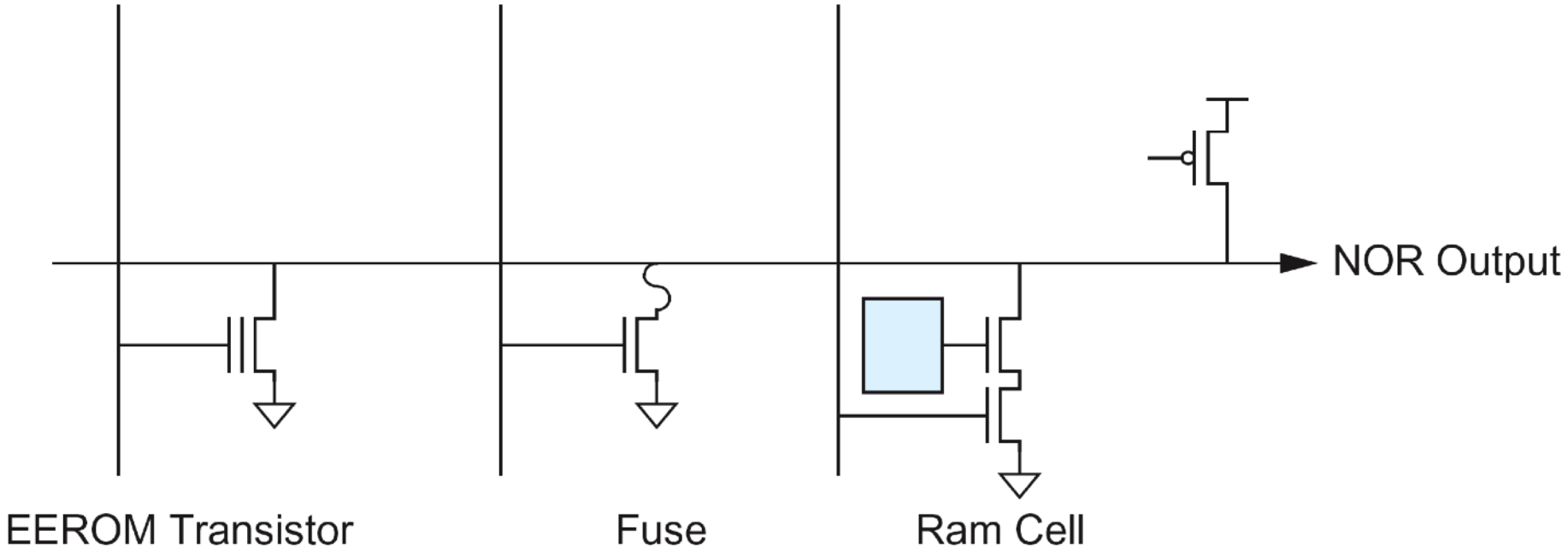


$$f_0 = x_0 x_1 + \overline{x_2}$$

$$f_1 = x_0 x_1 x_2 + \overline{x_2} + \overline{x_0} x_1$$

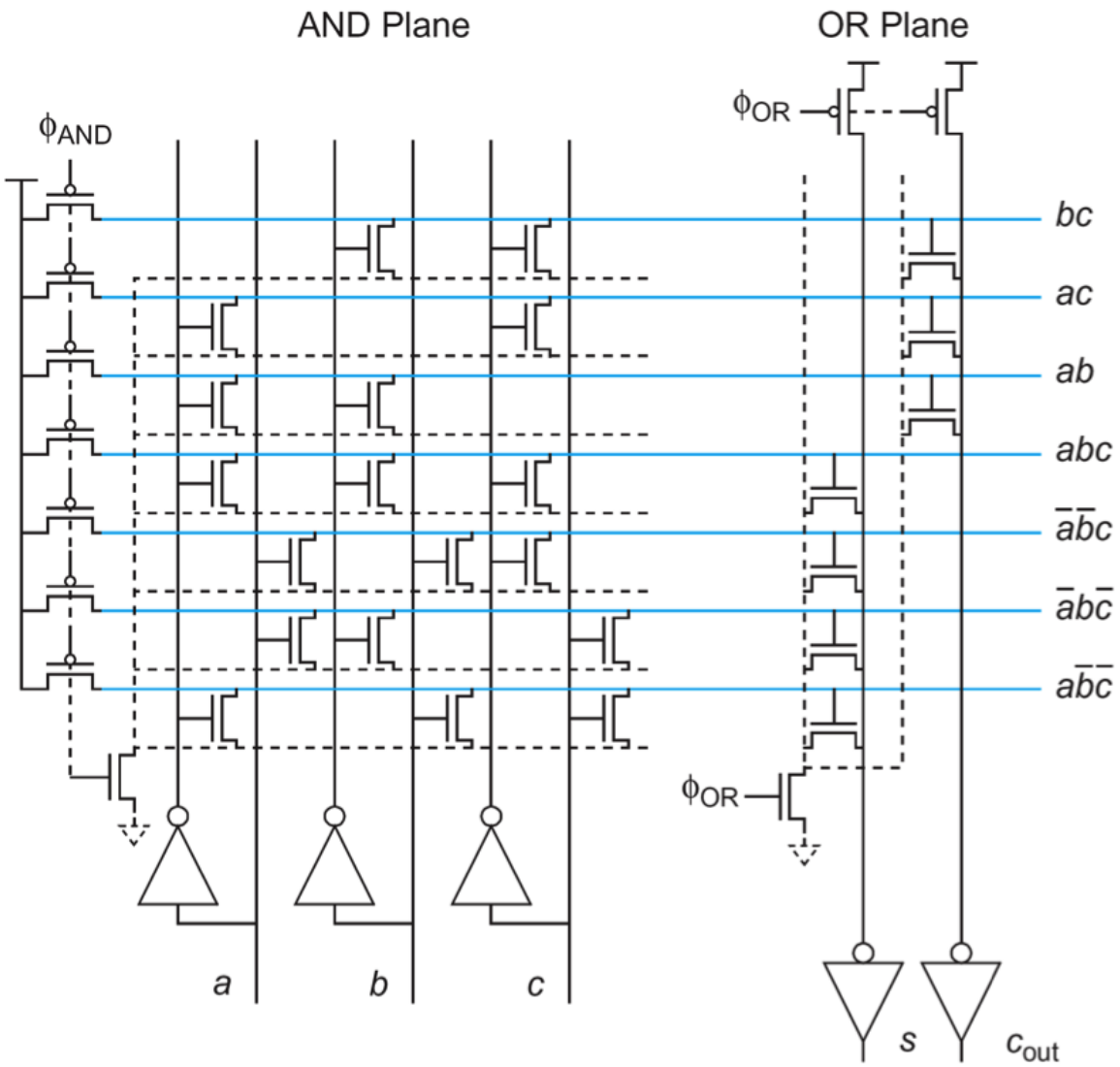
Adapted from [Rabaey'02]

Various Programmability Options for PLA/PROM NOR Structure



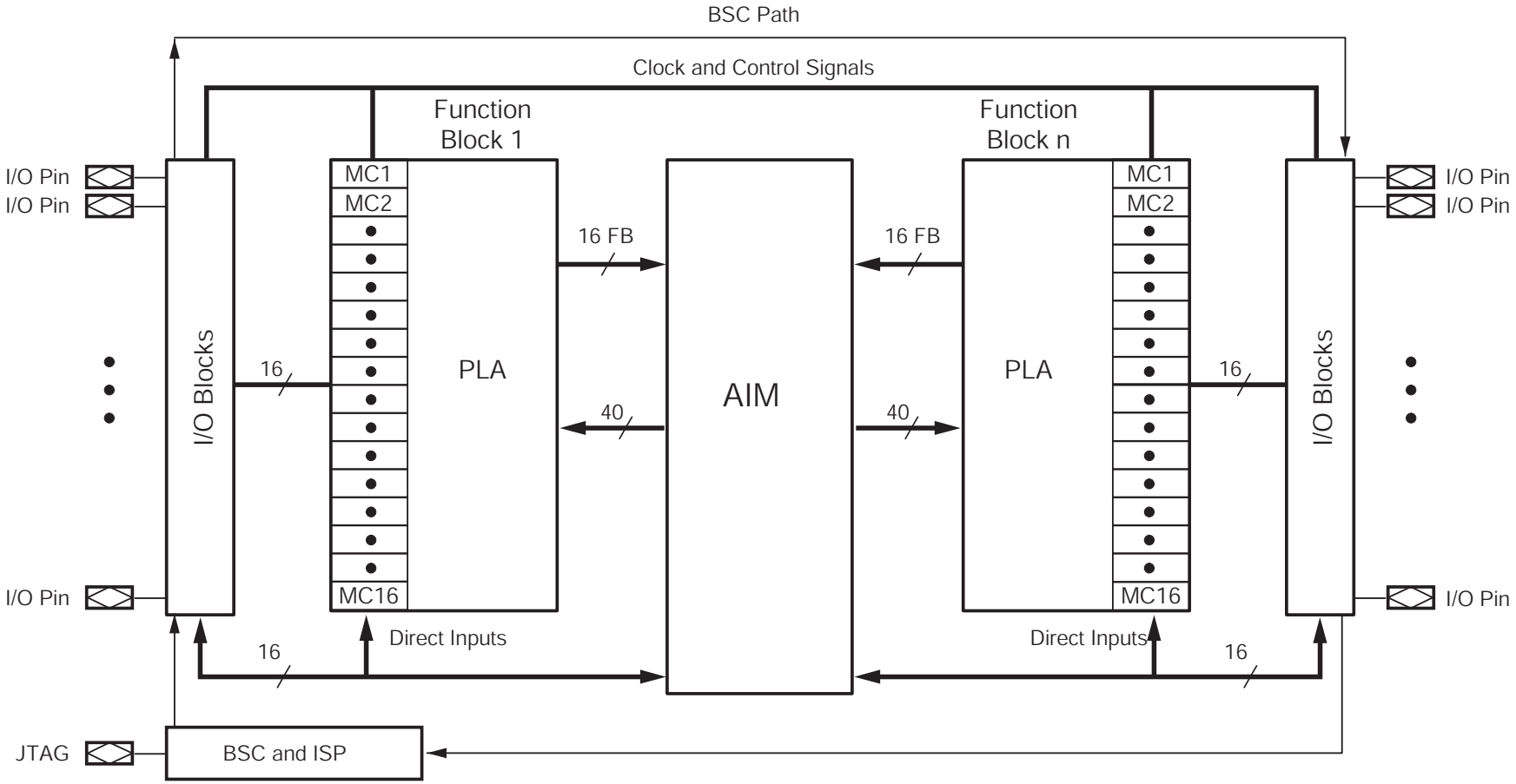
Adapted from [Weste'11]

PLA Implemented with Dynamic Logic



Adapted from [Weste'11]

Modern Complex Programmable Logic Devices



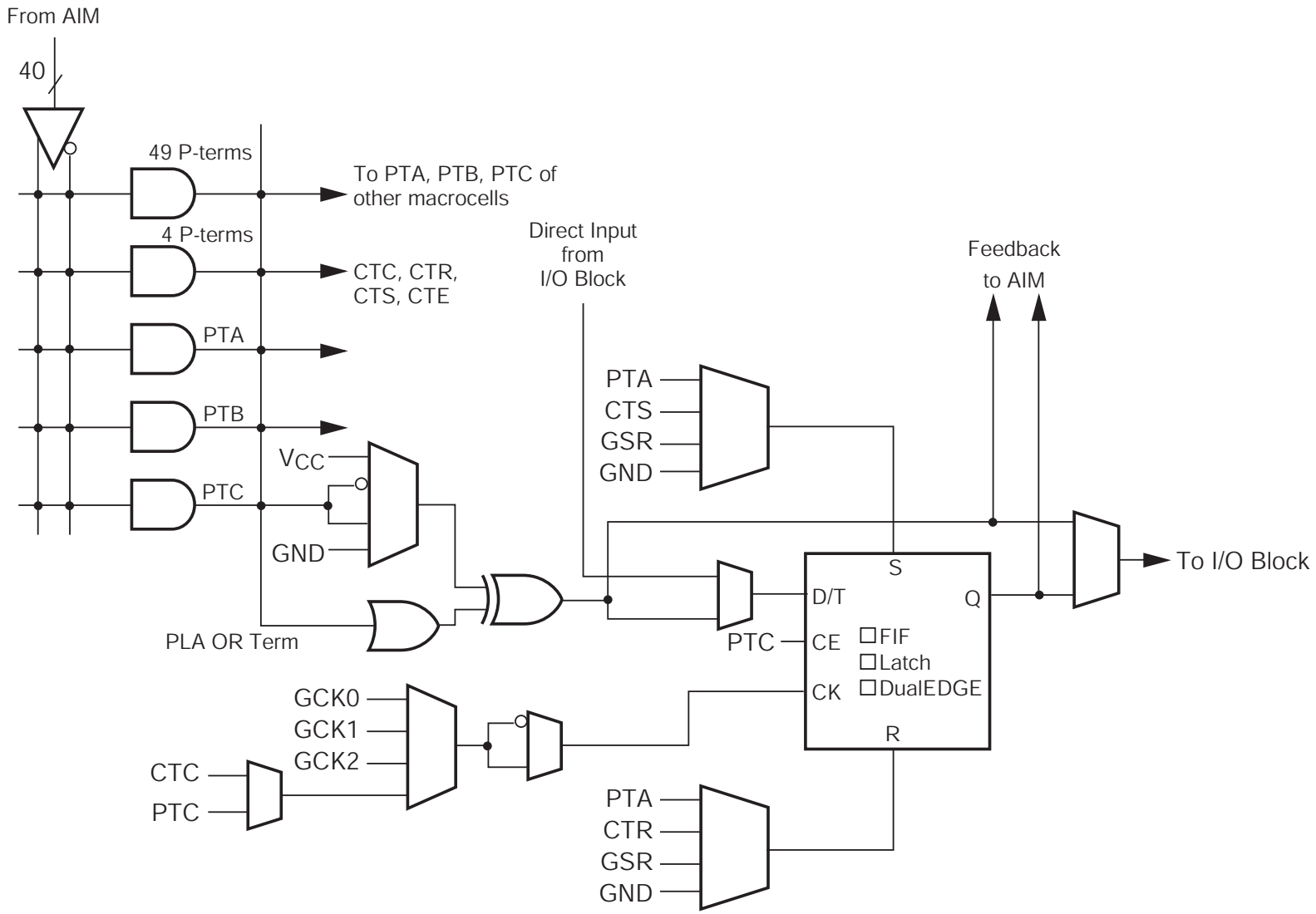
Xilinx CoolRunner-II CPLD: 2–32 PLAs integrated on a single chip

PLAs support 40 inputs, 56 product terms, and 16 output terms

PLAs chained together through a programmable “Advanced Interconnect Matrix”

Adapted from [Xilinx’08]

CPLD Macro Cell



Adapted from [Xilinx'08]

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

Gate-Array-Based Design

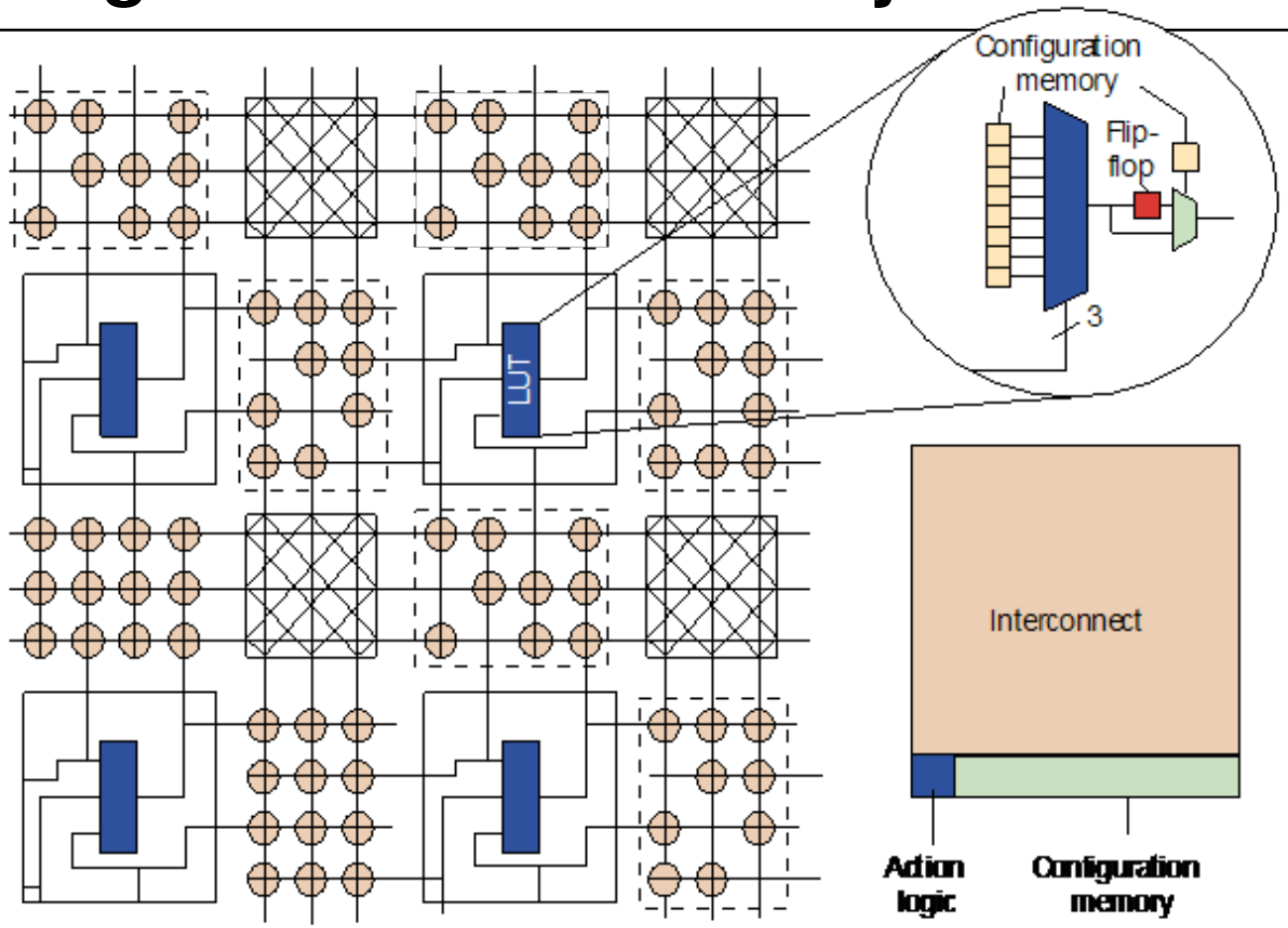
Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

Field-Programmable Gate-Arrays

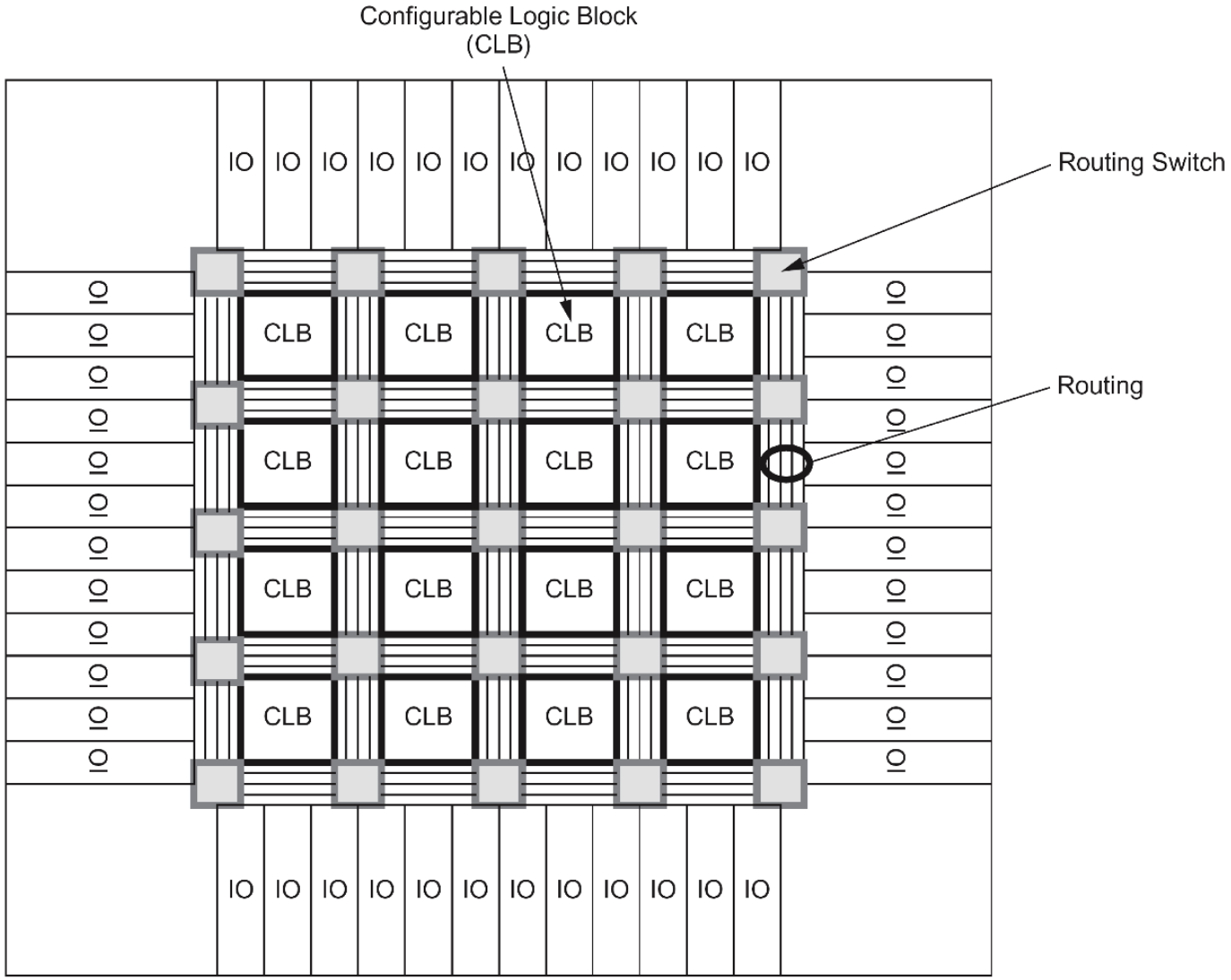
- Two-dimensional array of simple logic- and interconnection-blocks.
- Typical architecture: LUTs implement any function of n-inputs (n=3 in this case).
- Optional Flip-flop with each LUT.



- Fuses, EPROM, or Static RAM cells are used to store the "configuration".
  - Here, it determines function implemented by LUT, selection of Flip-flop, and interconnection points.
- Many FPGAs include special circuits to accelerate adder carry-chain and many special cores: RAMs, MAC, Enet, PCI, SERDES, ...

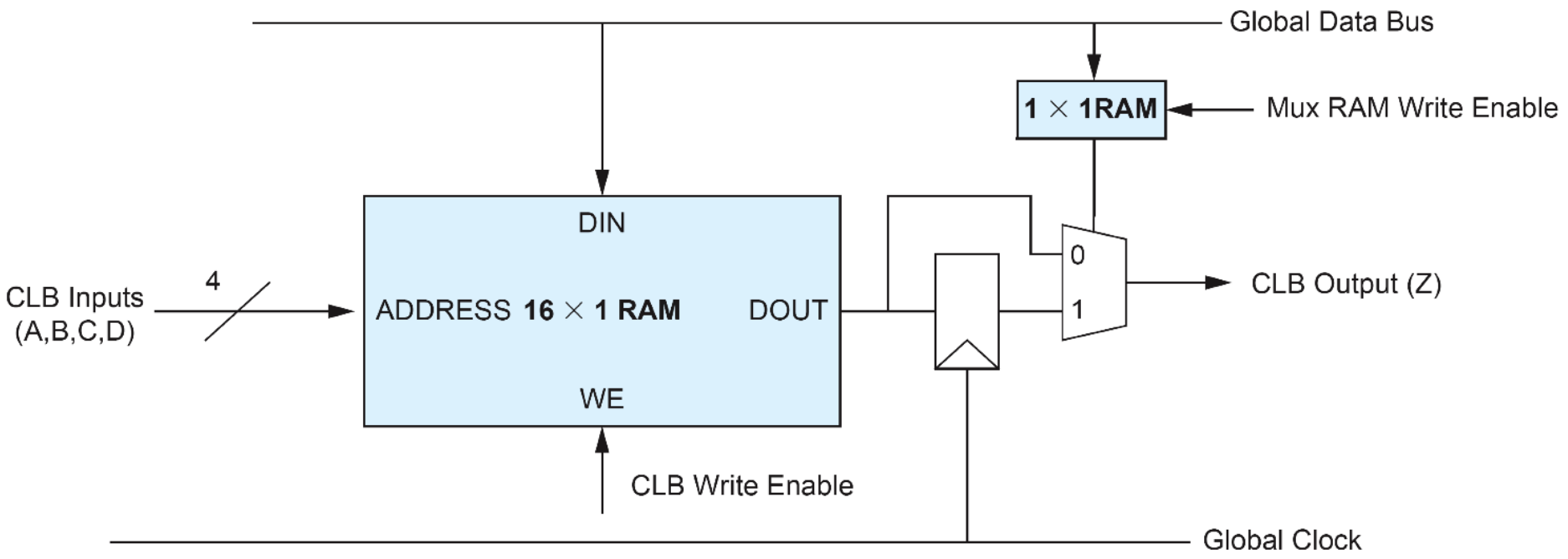
Adapted from [Asanovic'11]

Simplified FPGA Floorplan



Adapted from [Weste'11]

Simplified FPGA Combinational Logic Block



Adapted from [Weste'11]

Agenda

Standard-Cell-Based Design

System-on-Chip Platform-Based Design

Gate-Array-Based Design

Programmable Logic-Based Design

Reprogrammable Logic-Based Design

Comparison and Hybrids

Operation Binding Time

"Hardware"				"Software"	
Full Custom	Standard Cell	SoC Platform	Gate Array	Prog. Logic	Reprog. Logic	µproc DSP
First Mask	First Mask	First Mask	Metal Masks	Fuse Program	Load Config	Load Program

Later Binding Time

- Earlier the operation is bound, the less area, delay, and energy required for the implementation
- Later the operation is bound, the more flexible the device

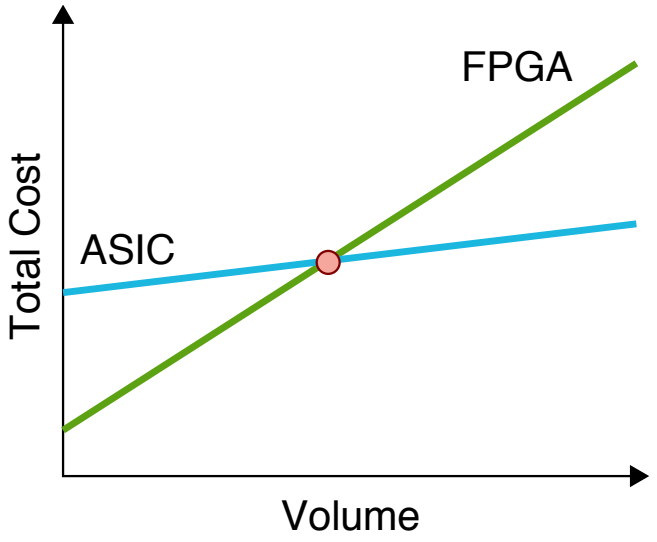
Adapted from [Asanovic'11]

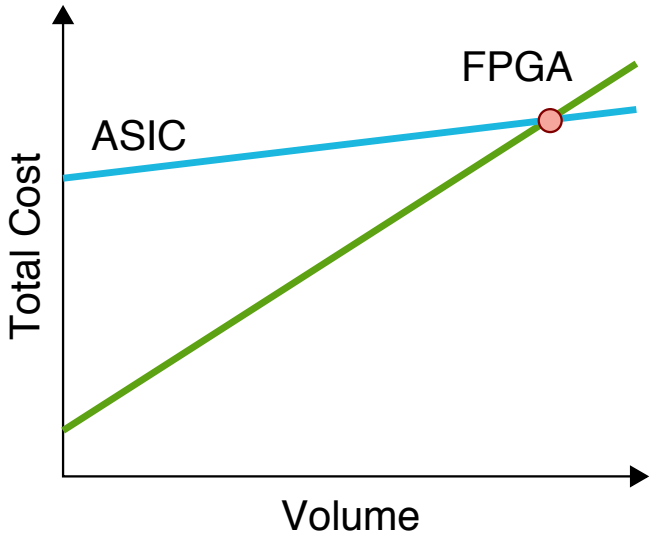
Comparison of Specific Design Methodologies

Design Method	NRE	Unit Cost	Power Disp	Impl Compl	Time to Market	Perf	Flex
Full Custom	VHigh	Low	Low	High	High	VHigh	Low
Standard Cell	High	Low	Low	High	High	High	Low
SoC Platform	High	Low	Low	Med	High	High	Med
Gate Array	Med	Med	Low	Med	Med	Med	Med
Prog Logic	Low	High	Med	Low	Low	Med	Med
Reprog Logic	Low	High	Med	Med	Low	High	High
µProc/DSP	Low	High	High	Low	Low	Low	VHigh

ASIC vs. FPGA

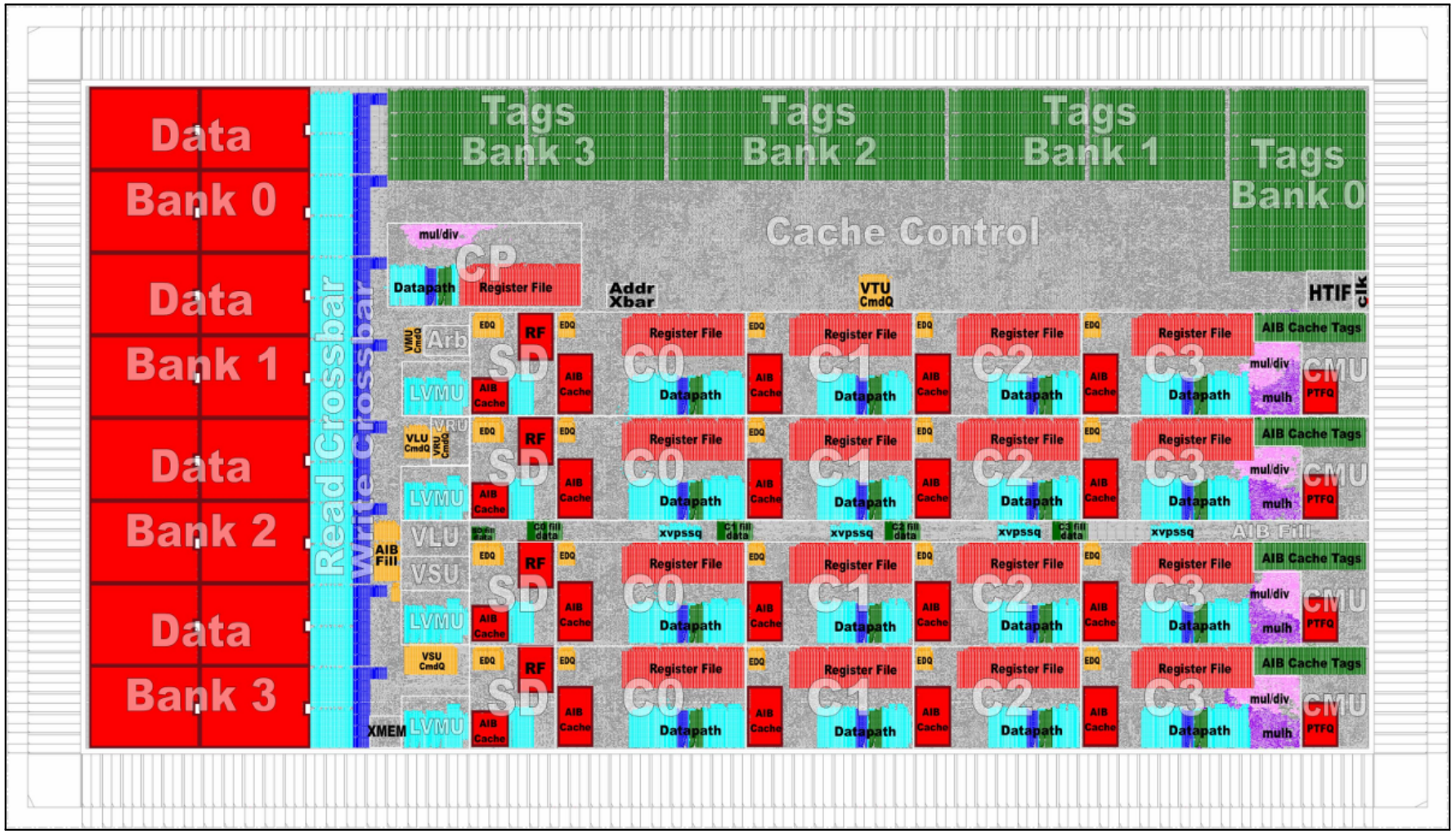
- Traditional Argument
  - ASIC: High NRE ($2M for 0.35 µm chip), low marginal cost, best effeciency
  - FPGA: Low NRE, high marginal cost, lower effeciency
  - Cross-over point: around 10,000
- Current Trends
  - ASIC: Increasing NRE ($40M for 90 nm chip) due to design costs, verification costs, mask costs, etc
  - FPGA: Better able to track Moore's law, integrating fixed function blocks
  - Cross-over point: around 100,000



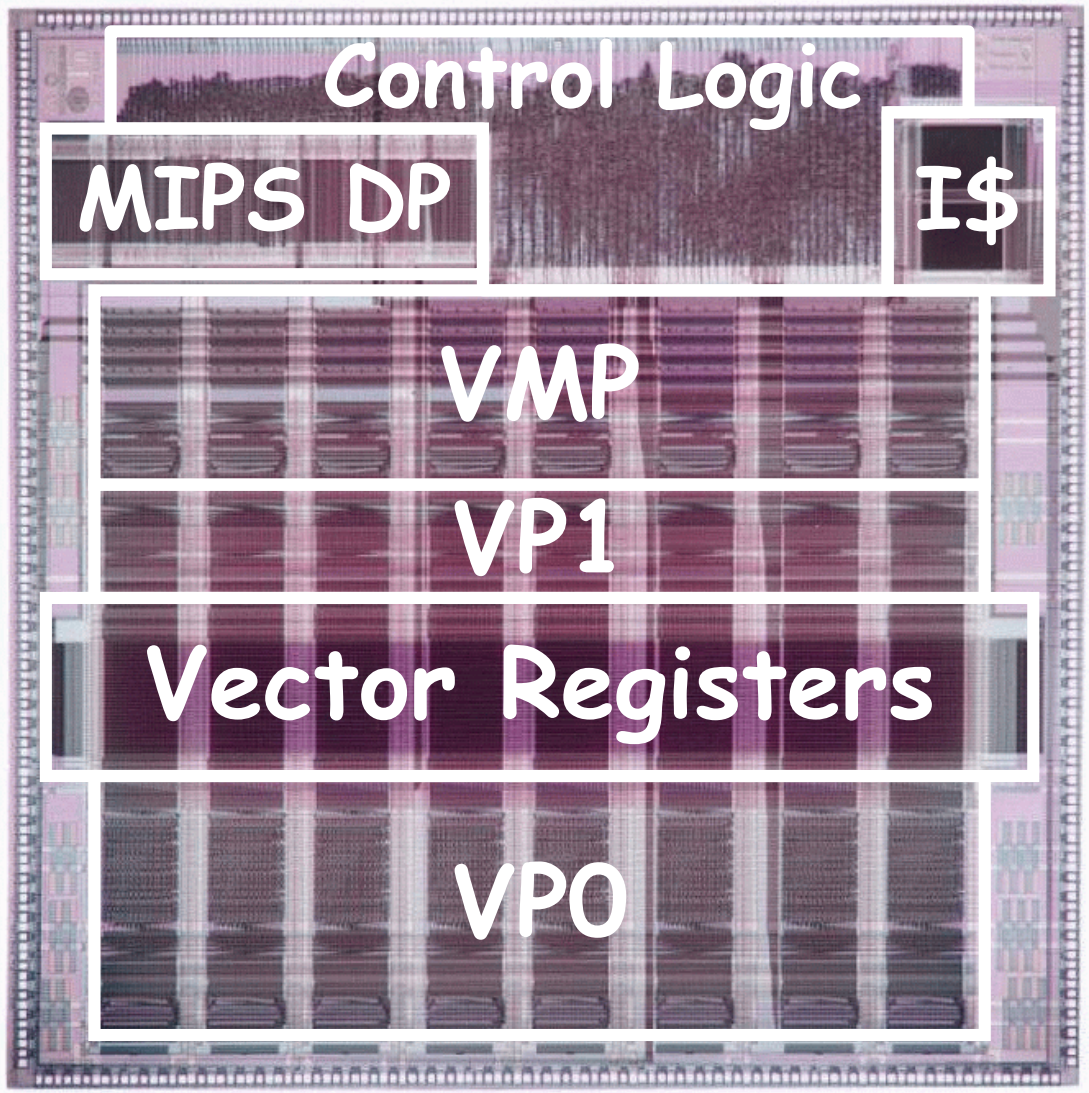


Adapted from [Asanovic'11]

Scale – ASIC with Pre-Placement & SRAMs



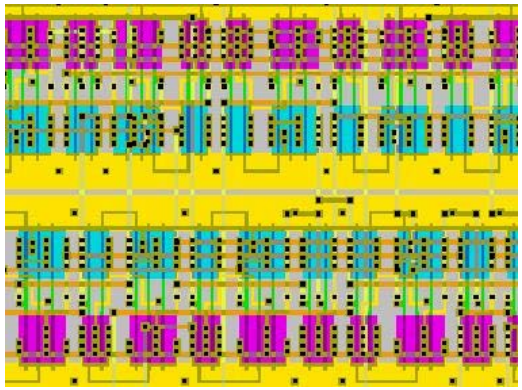
T0 – Full Custom w/ Standard Cells



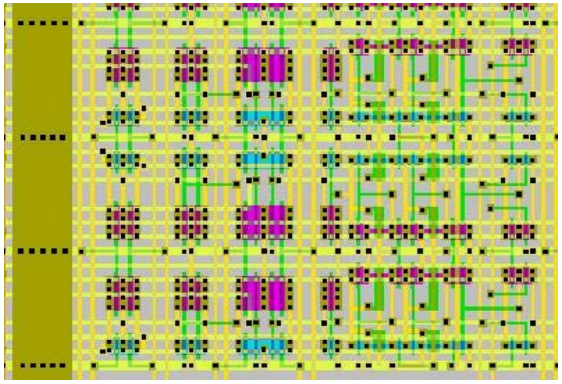
Adapted from [Terman'02]

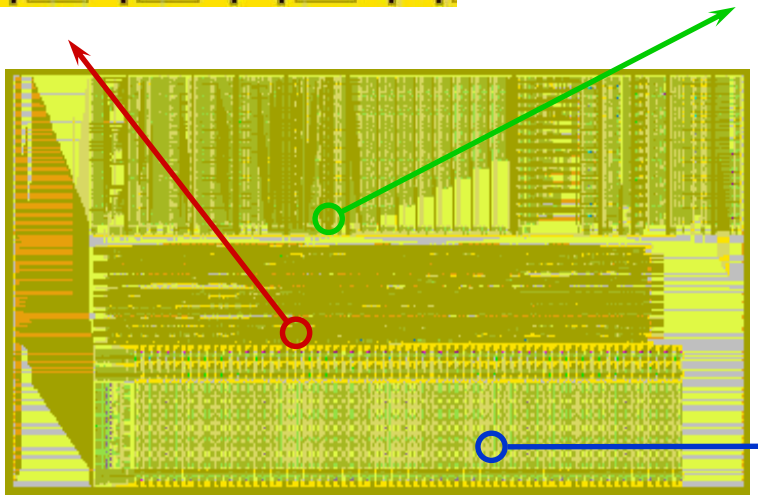
Application-Specific IC – Full Custom w/ Standard Cell

Standard cell: predefined gates, automatically placed and routed. In .5u → 10K fets/mm^2

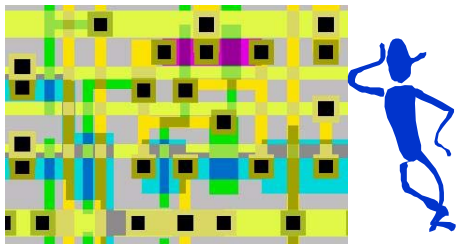


Full custom: custom "cells" meant to be stacked in columns to create N-bit wide datapath. Signals between columns routed across cells. In .5u → 25K/mm2



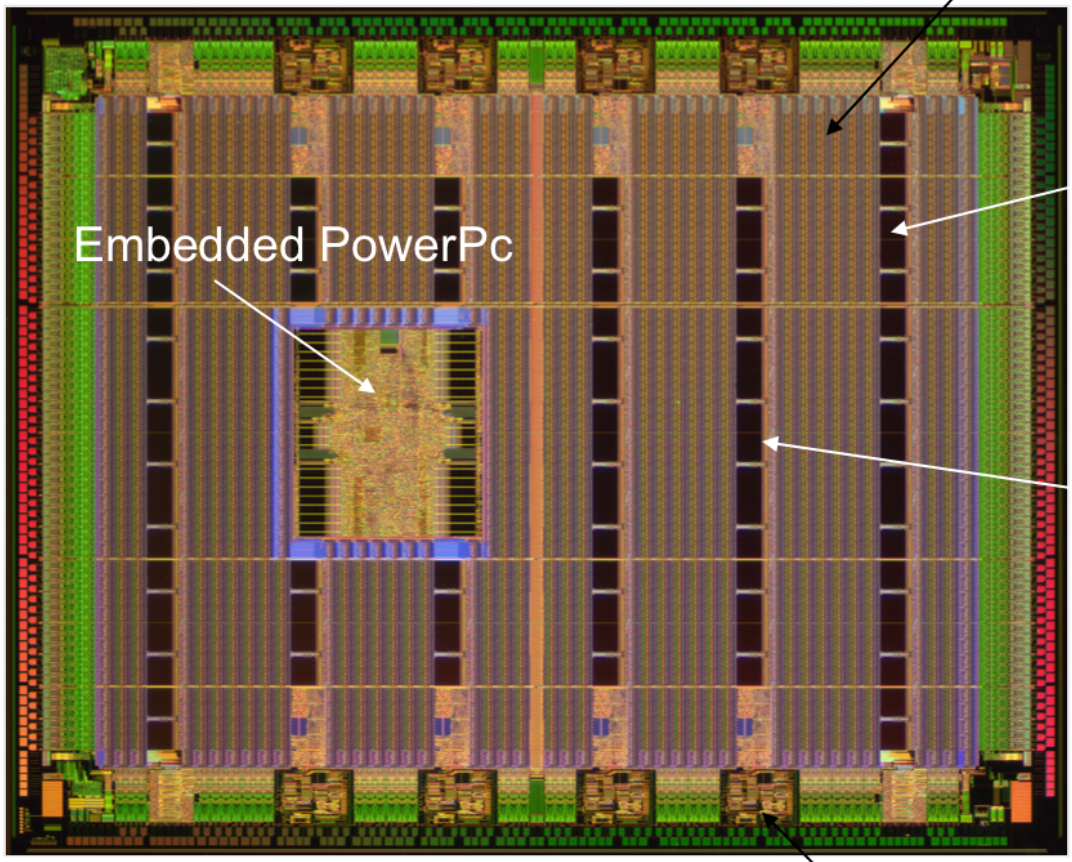


RAM Generator: one cell iterated many times perhaps surrounded by driver/sensing logic. Basic structure stays the same, only dimensions change. In .5u → 45K/mm^2 for multiport regfile



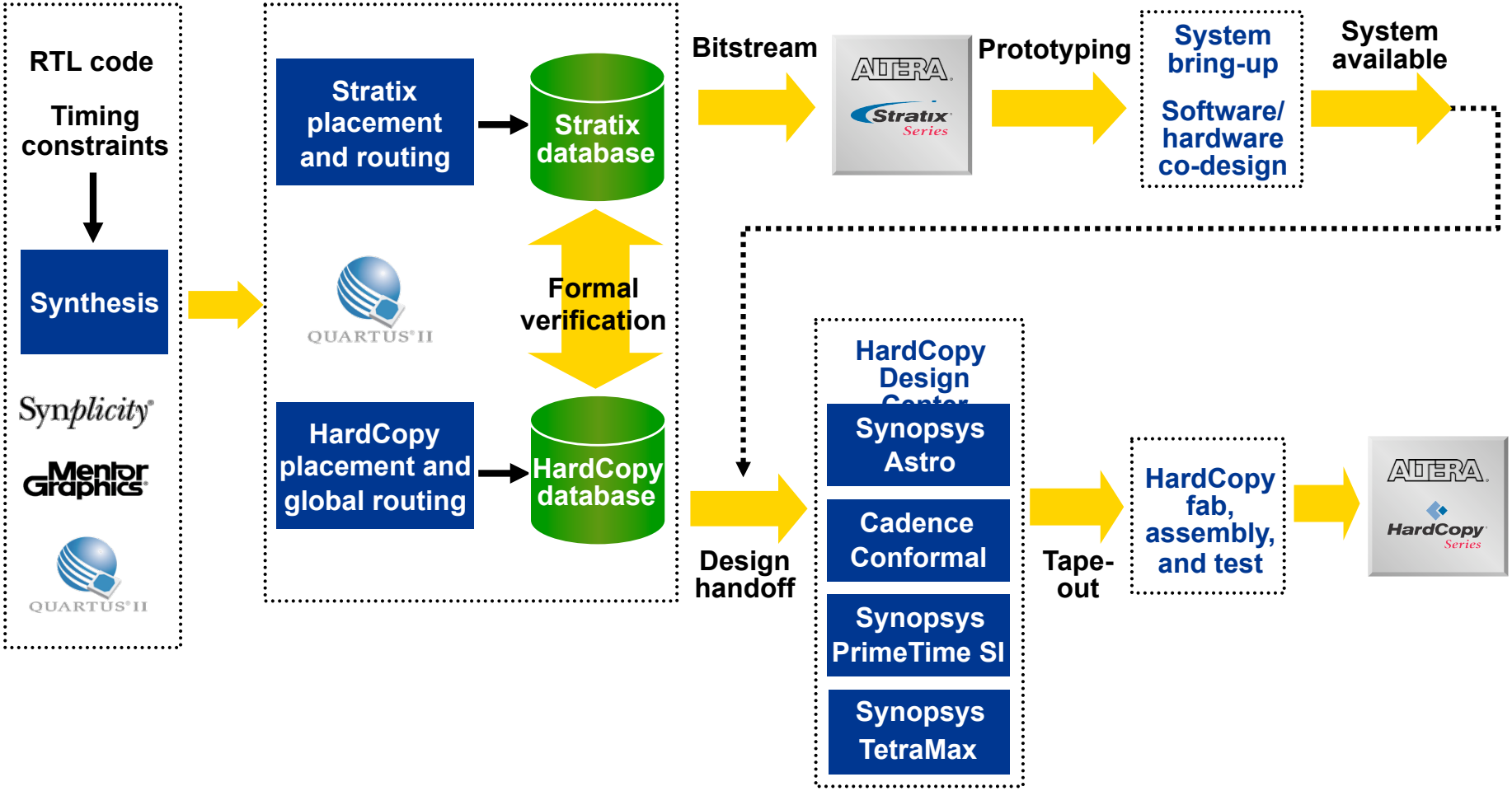
Adapted from [Terman'02]

Xilinx Vertex-II Pro – FPGA w/ Hard Processor



Adapted from [Rabaey'02]

Altera HardCopy – FPGA to Gate-Array-Like Tapeout



Adapted from [Mansur'08]

Acknowledgments

- [Asanovic'11] K. Asanović, J. Wawrzynek, and J. Lazzaro, UC Berkeley CS 250 VLSI Systems Design, Lecture Slides, 2011.
- [ASV'01] A.L. Sangiovanni-Vincentelli, "Platform-Based Design", UC Berkeley White Paper, 2001.
- [Mansur'08] D. Mansur, "Stratix IV FPGA and HardCopy IV ASIC @ 40 nm," Hot Chips, Aug. 2008.
- [Melikyan] V. Melikeyan, "IC Design Introduction", Lecture Slides, Synopsys Curriculum Materials.
- [Rabaey'02] J. Rabaey et al., Companion Slides for "Digital Integrated Circuits: Design Perspective," 2nd ed, Prentice Hall, 2002.

Acknowledgments

- [SAED'11] "Standard Cell Lib SAED_EDK90_CORE Databook", Synopsys, 2011.
- [Terman'02] C. Terman and K. Asanović, MIT 6.371 Introduction to VLSI Systems, Lecture Slides, 2002.
- [Weste'11] N. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 4th ed, Addison Wesley, 2011.
- [Xilinx'08] CoolRunner-II CPLD Family," Xilinx DataSheet, 2008. `http://www.xilinx.com/support/documentation/data_sheets/ds090.pdf`